To all our customers

Regarding the change of names mentioned in the document, such as Hitachi Electric and Hitachi XX, to Renesas Technology Corp.

The semiconductor operations of Mitsubishi Electric and Hitachi were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Hitachi, Hitachi, Ltd., Hitachi Semiconductors, and other Hitachi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Renesas Technology Home Page: http://www.renesas.com

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

Cautions

Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.

Hitachi I-ZTAT™ Microcomputer

H8/570

HD6475708, HD6435708

Hardware Manual

ADE-602-045
Rev. 1.0
6/2/00
Hitachi, Ltd.

Cautions

1. Hitachi neither warrants nor grants licenses of any rights of Hitachi's or any third party's patent, copyright, trademark, or other intellectual property rights for information contained in this document. Hitachi bears no responsibility for problems that may arise with third party's rights, including intellectual property rights, in connection with use of the information contained in this document.
2. Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.
3. Hitachi makes every attempt to ensure that its products are of high quality and reliability. However, contact Hitachi's sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.
4. Design your application so that the product is used within the ranges guaranteed by Hitachi particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Hitachi bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Hitachi product does not cause bodily injury, fire or other consequential damage due to operation of the Hitachi product.
5. This product is not designed to be radiation resistant.
6. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Hitachi.
7. Contact Hitachi's sales office for any questions regarding this document or Hitachi semiconductor products.

Preface

The H8/570 is a continuation of Hitachi's ZTAT™* microcomputer line, and its concept differs uniquely from conventional single-chip microcomputers. The core of the H8/570 is a user-programmable module called an ISP (Intelligent Sub-Processor).

The ISP can implement a wide variety of peripheral functions through instruction execution of user programs. Since the ISP has an EPROM-based memory, the user can easily program ISP instructions, similar to conventional ZTAT™ microcomputers. Consequently, the H8/570 provides high flexibility for the user.

The H8/570 is quick and flexible to manage:

- Redesign work
- Quick development and trial production
- Specified alterations in the early stages of mass production

In addition to the ISP, the H8/570 has a high-performance H8/500 CPU and several on-chip support functions such as mass storage RAM, a PWM (pulse width modulation) timer, a serial communication interface (SCI), an A/D converter, and I/O ports. Since the H8/570 can easily achieve high performance with its compact design, the H8/570 is suitable as an embedded-type controller for various applications.

This manual gives only the hardware description of the H8/570, and excludes the explanation for the ISP. For details on the ISP, refer to the *ISP Hardware Manual* and the *ISP Programming Manual*. For details on the CPU instruction set, refer to the *H8/500 Series Programming Manual*, which applies to all chips in the H8/500 Series.

Note: * ZTAT™ (Zero Turn Around Time) is a trademark of Hitachi, Ltd.

Contents

Section 1 Overview 1
1.1 Features 1
1.2 Block Diagram 5
1.3 Pin Assignments and Functions 6
1.3.1 Pin Arrangement 6
1.3.2 Pin Functions 7

Section 2 MCU Operating Modes and Address Space 15
2.1 Overview 15
2.2 Mode Descriptions 16
2.3 Address Map 16
2.4 Mode Control Register (MDCR) 18

Section 3 CPU 19
3.1 Overview 19
3.1.1 Features 19
3.1.2 Address Space 20
3.1.3 Register Configuration 21
3.2 CPU Register Descriptions 22
3.2.1 General Registers 22
3.2.2 Control Registers 23
3.2.3 Initial Register Values 26
3.3 Data Formats 28
3.3.1 Data Formats in General Registers 28
3.3.2 Data Formats in Memory 29
3.4 Instructions 30
3.4.1 Basic Instruction Formats 30
3.4.2 Addressing Modes 31
3.5 Instruction Set 35
3.5.1 Overview 35
3.5.2 Data Transfer Instructions 37
3.5.3 Arithmetic operations 38
3.5.4 Logic Operation Instructions 39
3.5.5 Shift Operation Instructions 40
3.5.6 Bit Manipulation Instructions 40
3.5.7 Branching Instructions 41
3.5.8 System Control Instructions 42
3.5.9 Short-Format Instructions 45
3.6 Operating Modes 45

3.6.1 Minimum Mode 45
3.6.2 Maximum Mode 46
3.6.3 8-Bit Data Bus Mode 46
3.6.4 16-Bit Data Bus Mode 46
3.7 Basic Operational Timing 46
3.7.1 Overview 46
3.7.2 Access to Memory (RAM) 47
3.7.3 Access to Register Field 49
3.7.4 Access to External Three-State Access Space 51
3.7.5 Access to External Two-State Word-Wide Space 53
3.8 CPU States 55
3.8.1 Overview 55
3.8.2 Program Execution State 56
3.8.3 Exception-Handling State 56
3.8.4 Bus-Released State 57
3.8.5 Reset State 58
3.8.6 Power-Down State 58

Section 4 Exception Handling 63
4.1 Overview 63
4.1.1 Types of Exception Handling and Their Priority 63
4.1.2 Hardware Exception-Handling Sequence 64
4.1.3 Exception Factors and Vector Table 65
4.2 Reset 67
4.2.1 Overview 67
4.2.2 Reset Sequence 67
4.2.3 Stack Pointer Initialization 68
4.3 Address Error 71
4.3.1 Illegal Instruction Prefetch 71
4.3.2 Word-Size Data Access at an Odd Address 71
4.4 Trace 72
4.5 Interrupts 72
4.6 Invalid Instruction 73
4.7 Trap Instructions and Zero Divide 73
4.8 Cases in which Exception Handling is Deferred 74
4.8.1 Instructions that Disable Interrupts 74
4.8.2 Disabling of Exceptions Immediately after Reset 75
4.8.3 Disabling of Interrupts after a Data Transfer Cycle 75
4.9 Stack Status after Completion of Exception Handling 76
4.9.1 PC Value Pushed Onto Stack for Trace, Interrupts, Trap Instructions, and Zero Divide Exceptions 77
4.9.2 PC Value Pushed Onto Stack for Address Error and Invalid Instruction Exceptions 77

4.10 Notes on Use of the Stack 77

Section 5 Interrupt Controller 79
5.1 Overview 79
5.1.1 Features 79
5.1.2 Block Diagram 80
5.1.3 Register Configuration 81
5.2 Interrupt Types 81
5.2.1 External Interrupts 81
5.2.2 Internal Interrupts 82
5.2.3 Interrupt Vector Table 82
5.3 Register Descriptions 84
5.3.1 Interrupt Priority Registers A to D (IPRA to IPRD) 84
5.3.2 Timing of Priority Setting 85
5.4 Interrupt Handling Sequence 85
5.4.1 Interrupt Handling Flow 85
5.4.2 Stack Status After Interrupt Handling Sequence 88
5.4.3 Timing of Interrupt Exception-Handling Sequence 90
5.5 Interrupts During Operation of the Data Transfer Controller 92
5.6 Interrupt Response Time 93

Section 6 Data Transfer Controller 95
6.1 Overview 95
6.1.1 Features 95
6.1.2 Block Diagram 95
6.1.3 Register Configuration 96
6.2 Register Descriptions 97
6.2.1 Data Transfer Mode Register (DTMR) 97
6.2.2 Data Transfer Source Address Register (DTSR) 98
6.2.3 Data Transfer Destination Register (DTDR) 98
6.2.4 Data Transfer Count Register (DTCR) 98
6.2.5 Data Transfer Enable Registers A to D (DTEA to DTED) 99
6.3 Data Transfer Operation 100
6.3.1 Data Transfer Cycle 100
6.3.2 DTC Vector Table 102
6.3.3 Location of Register Information in Memory 104
6.3.4 Length of Data Transfer Cycle 104
6.4 Procedure for Using the DTC 106
6.5 Example of Using the DTC 106

Section 7 Wait-State Controller 109
7.1 Overview 109
7.1.1 Features 109

7.1.2 Block Diagram 110
7.1.3 Register Configuration 110
7.2 Wait-State Control Register 111
7.3 Operation in Each Wait Mode 112
7.3.1 Programmable Wait Mode 113
7.3.2 Pin Wait Mode 114
7.3.3 Pin Auto-Wait Mode 115

Section 8 Clock Pulse Generator 117
8.1 Overview 117
8.1.1 Block Diagram 117
8.2 Oscillator Circuit 117
8.3 System Clock Divider 120

Section 9 I/O Ports 121
9.1 Overview 121
9.2 Port 1 123
9.2.1 Overview 123
9.2.2 Port 1 Registers 123
9.2.3 Pin Functions in Each Mode 126
9.3 Port 5 128
9.3.1 Overview 128
9.3.2 Port 5 Registers 128
9.3.3 Pin Functions in Each Mode 130
9.3.4 Built-In MOS Pull-Up 131
9.4 Port 6 132
9.4.1 Overview 132
9.4.2 Port 6 Registers 132
9.4.3 Pin Functions in Each Mode 134
9.5 Port 7 136
9.5.1 Overview 136
9.5.2 Port 7 Register 136
9.6 Port 8 137
9.6.1 Overview 137
9.6.2 Port 8 Registers 137
9.7 Port 9 141
9.7.1 Overview 141
9.7.2 Port 9 Registers 141
9.8 Port 10 145
9.8.1 Overview 145
9.8.2 Port 10 Registers 145
9.9 Port 11 149
9.9.1 Overview 149

9.9.2 Port 11 Registers 149
9.10 Port 12 151
9.10.1 Overview 151
9.10.2 Port 12 Registers 151
9.10.3 Pin Functions in Each Mode 153

Section 10 PWM (Pulse Width Modulation) Timer 155
10.1 Overview 155
10.1.1 Features 155
10.1.2 Block Diagram 156
10.1.3 Output Pins 157
10.1.4 Register Configuration 157
10.2 Register Descriptions 158
10.2.1 Timer (TMR)—H'FEAC, H'FEAD 158
10.2.2 Output Compare Registers 0 to 2 (OCR0 to OCR2)—H'FEA6 and H'FEA7, H'FEA8 and H'FEA9, H'FEAA and H'FEAB 158
10.2.3 Output Data Registers 0 to 2 (ODR0 to ODR2)—H'FEA3, H'FEA4, H'FEA5 159
10.2.4 Output Data Latch (ODL)—H'FEA2 160
10.2.5 Timer Control Register (TCR)—H'FEA0 160
10.2.6 Timer Status Register (TMSR)—H'FEA1 162
10.3 PWM Operating Timing 165
10.4 Interface with CPU and ISP 167
10.4.1 Read or Write of TMR and OCRs 167
10.4.2 Write to TMR 167
10.4.3 Write to OCRs 168
10.5 CPU Interrupts and DTC Interrupts 168
10.6 Data Transfer by ISP 169
10.7 Sample Application 169
10.7.1 Six-Phase Pulse Output 169
10.7.2 Independent Waveform Output 171
10.8 Application Notes 172
10.8.1 Incrementation Caused by Changing the Internal Clock Source 172

Section 11 Watchdog Timer 175
11.1 Overview 175
11.1.1 Features 175
11.1.2 Block Diagram 176
11.1.3 Register Configuration 176
11.2 Register Descriptions 177
11.2.1 Timer Counter (TCNT)—H'FE8B 177
11.2.2 Timer Control/Status Register (TCSR)—H'FE8A (Read), H'FE8B (Write) 177
11.2.3 Reset Control/Status Register (RSTCSR)—H'FF4F 179
11.2.4 Notes on Register Access 180

11.3 Operation 182
11.3.1 Watchdog Timer Mode 182
11.3.2 Interval Timer Mode 183
11.3.3 Operation in Software Standby Mode 184
11.3.4 Setting of Overflow Flag 184
11.3.5 Setting of Watchdog Timer Reset Bit 185
11.4 Application Notes 186

Section 12 Serial Communication Interface 187
12.1 Overview 187
12.1.1 Features 187
12.1.2 Block Diagram 188
12.1.3 Input and Output Pins 188
12.1.4 Register Configuration 189
12.2 Register Descriptions 189
12.2.1 Receive Shift Register (RSR) 189
12.2.2 Receive Data Register (RDR)—H'FE9D 189
12.2.3 Transmit Shift Register (TSR) 190
12.2.4 Transmit Data Register (TDR)—H'FE9B 190
12.2.5 Serial Mode Register (SMR)—H'FE98 190
12.2.6 Serial Control Register (SCR)—H'FE9A 192
12.2.7 Serial Status Register (SSR)—H'FE9C 194
12.2.8 Bit Rate Register (BRR)—H'FE99 196
12.3 Operation 201
12.3.1 Overview 201
12.3.2 Asynchronous Mode 202
12.3.3 Synchronous Mode 206
12.4 CPU Interrupts and DTC Interrupts 209
12.5 Application Notes 210

Section 13 A/D Converter 213
13.1 Overview 213
13.1.1 Features 213
13.1.2 Block Diagram 214
13.1.3 Input Pins 215
13.2 Register Descriptions 216
13.2.1 A/D Data Registers (ADDR)—H'FE80 to H'FE87 216
13.2.2 A/D Control/Status Register (ADCSR)—H'FE88 217
13.2.3 A/D Control Register (ADCR)—H'FE89 219
13.3 CPU Interface 220
13.4 Operation 221
13.4.1 Single Mode 221
13.4.2 Scan Mode 224

13.4.3 Input Sampling Time and A/D Conversion Time 226
13.4.4 A/D Conversion Start by External Trigger Input 227
13.5 Interrupts and the Data Transfer Controller 227

Section 14 RAM 229
14.1 Overview 229
14.1.1 Block Diagram 229
14.1.2 Register Configuration 230
14.2 RAM Control Register (RAMCR) 230
14.3 Operation 231

Section 15 ISP Program Memory 233
15.1 PROM Mode 233
15.1.1 PROM Mode Setup 233
15.1.2 Socket Adapter Pin Assignments and Memory Map 233
15.2 Programming 236
15.2.1 Writing and Verifying 236
15.2.2 Reliability of Written Data 240

Section 16 Power-Down State 241
16.1 Overview 241
16.2 Sleep Mode 242
16.2.1 Entering Sleep Mode 242
16.2.2 Exiting Sleep Mode 242
16.3 Software Standby Mode 242
16.3.1 Entering Software Standby Mode 242
16.3.2 Software Standby Control Register (SBYCR) 243
16.3.3 Exiting Software Standby Mode 244
16.3.4 Application Notes 244
16.4 Hardware Standby Mode 244
16.4.1 Entering Hardware Standby Mode 244
16.4.2 Exiting Hardware Standby Mode 245
16.4.3 Timing Sequence of Hardware Standby Mode 245

Section 17 E-Clock Interface 247
17.1 Overview 247

Section 18 Electrical Specifications 251
18.1 Absolute Maximum Ratings 251
18.2 Electrical Characteristics 252
18.2.1 DC Characteristics 252
18.2.2 AC Characteristics 255
18.2.3 A/D Converter Characteristics 260

18.3 MCU Operational Timing 261
18.3.1 Bus Timing 261
18.3.2 Control Signal Timing 265
18.3.3 Clock Timing 266
18.3.4 I/O Port Timing 268
18.3.5 Pulse Width Modulation Timer Timing 268
18.3.6 Serial Communication Interface Timing 268
18.3.7 Intelligent Sub-Processor Interface Timing 269

Appendix A Instructions 271
A.1 Instruction Set 271
A.2 Instruction Codes 278
A.3 Operation Code Maps 293
A.4 Instruction Execution Cycles 298

Appendix B Register Field 308
B.1 Register Addresses and Bit Names 308
B.2 Register Descriptions 317

Appendix C I/O Port Schematic Diagrams 348
C.1 Schematic Diagram of Port 1 348
C.2 Schematic Diagram of Port 5 353
C.3 Schematic Diagram of Port 6 357
C.4 Schematic Diagram of Port 7 361
C.5 Schematic Diagram of Port 8 362
C.6 Schematic Diagram of Port 9 363
C.7 Schematic Diagram of Port 10 365
C.8 Schematic Diagram of Port 11 366
C.9 Schematic Diagram of Port 12 367

Appendix D Memory Map 368

Appendix E Pin State 369
E.1 Port State of Each Pin State 369
E.2 Pin Status in the Reset State 372

Appendix F Package Dimensions 375

Appendix G Abbreviations Listing 376

Section 1 Overview

1.1 Features

The H8/570, from Hitachi's ZTAT™ microcomputer line, was designed with a different concept which sets it apart from conventional microcomputers.

The core of the H8/570 is a user-programmable ISP (Intelligent Sub-Processor). This ISP uses a dedicated instruction set and includes components such as an execution unit. In the past, peripheral functions such as timers and communication functions have been implemented only by dedicated components. However, the ISP can quickly implement these functions through high-speed instruction execution which utilizes the same components to perform several functions.

Since the ISP has an EPROM-based memory, users are able to program the ISP. Therefore, the ISP can achieve not only standard functions but also functions customized for individual applications. When the user's design work is finished, the programs can be written into the H8/570 by using a standard EPROM programmer.

In addition to the ISP, the H8/570 incorporates Hitachi's original microcomputer H8/500 CPU and several on-chip support functions such as a serial communication interface (SCI), a PWM (pulse width modulation) timer, an A/D converter, a watchdog timer, I/O ports, and 2 kbytes of RAM.

Table 1.1 Features

Feature	Description
CPU	• General-register machine — Eight 16-bit general registers — Five 8-bit and two 16-bit control registers • High speed — Maximum clock rate: 12 MHz (24-MHz oscillator frequency) • Expanded operating modes supporting external memory — Minimum mode: Up to 64-kbyte address space — Maximum mode: Up to 1-Mbyte address space • Highly orthogonal instruction set — Addressing modes and data size can be specified independently for each instruction • 1.5 addressing modes — Register-register operations — Register-memory operations • Instruction set optimized for C language — Special short formats for frequently-used instructions and addressing modes
Memory	• 2-kbyte on-chip high-speed RAM
ISP (Intelligent Sub-Processor)	• Independent execution of 12 functions — Programmable function execution sequence — Event-driven execution by external event inputs • High-speed instruction execution — Execution time of one instruction: 84 ns (12-MHz clock rate) • Abundant internal registers — Thirty-two 16-bit internal data registers • Abundant ISP flags — 24 I/O flags — 16 interrupt status flags — 16 interconnection flags — Eight edge-detection flags • Bus master operation by the ISP • Instruction capacity: 512 words × 64 bits

Feature	Description
PWM timer	• Six output pins • Each output pin can independently generate pulses having a duty cycle ratio from 0 to 100% • Counter cycle can be set (16-bit length maximum) • Maximum resolution: 16 bits
Watchdog timer (WDT) (1 channel)	• Reset by overflow • Selectable as an interval timer
Serial communication interface (SCI)	• Selectable as asynchronous or synchronous mode • Full duplex: Simultaneous sending and receiving possible • Built-in baud rate generator
A/D converter	• 10-bit resolution • Eight channels controlled in either selected single mode or scan mode • Sample-and-hold function • External trigger to start A/D conversion
I/O ports	• 58 input/output pins (five 8-bit ports, one 6-bit port, one 7-bit port, one 5-bit port) • Eight input-only pins (one 8-bit port)
Interrupt controller (INTC)	• Two external interrupt pins (NMI, $\overline{IRQ}_0$) • 24 internal interrupts • Eight priority levels
Data transfer controller (DTC)	• Bidirectional data transfer between memory and I/O independent of the CPU
Wait-state controller (WSC)	• Wait state insertion during access to external memory or I/O

Feature	Description
Operating modes	• Four MCU operating modes — 8-bit expanded minimum mode, supporting up to 64-kbyte external memory (Mode 4) — 16-bit expanded minimum mode, supporting up to 64-kbyte external memory (Mode 1) — 8-bit expanded maximum mode, supporting up to 1-Mbyte external memory (Mode 6) — 16-bit expanded maximum modes, supporting up to 1-Mbyte external memory (Modes 3 and 5) • Three power-down modes — Sleep mode — Software standby mode — Hardware standby mode
Other features	• E-clock output available • On-chip clock generator • Packages — 112-pin QFP (FP-112) for HD6475708F and HD6435708F

1.2 Block Diagram

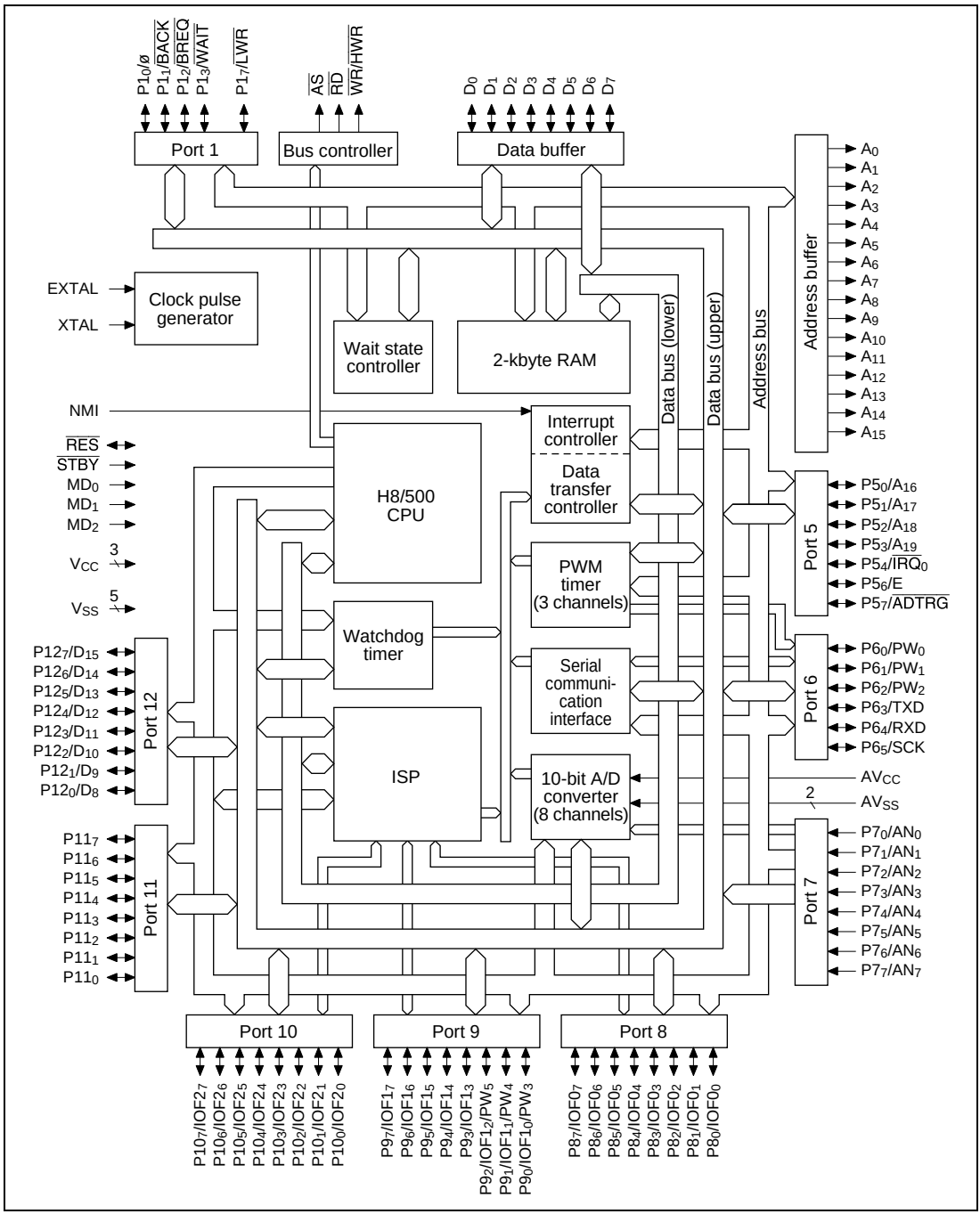


Figure 1.1 H8/570 Block Diagram

1.3 Pin Assignments and Functions

1.3.1 Pin Arrangement

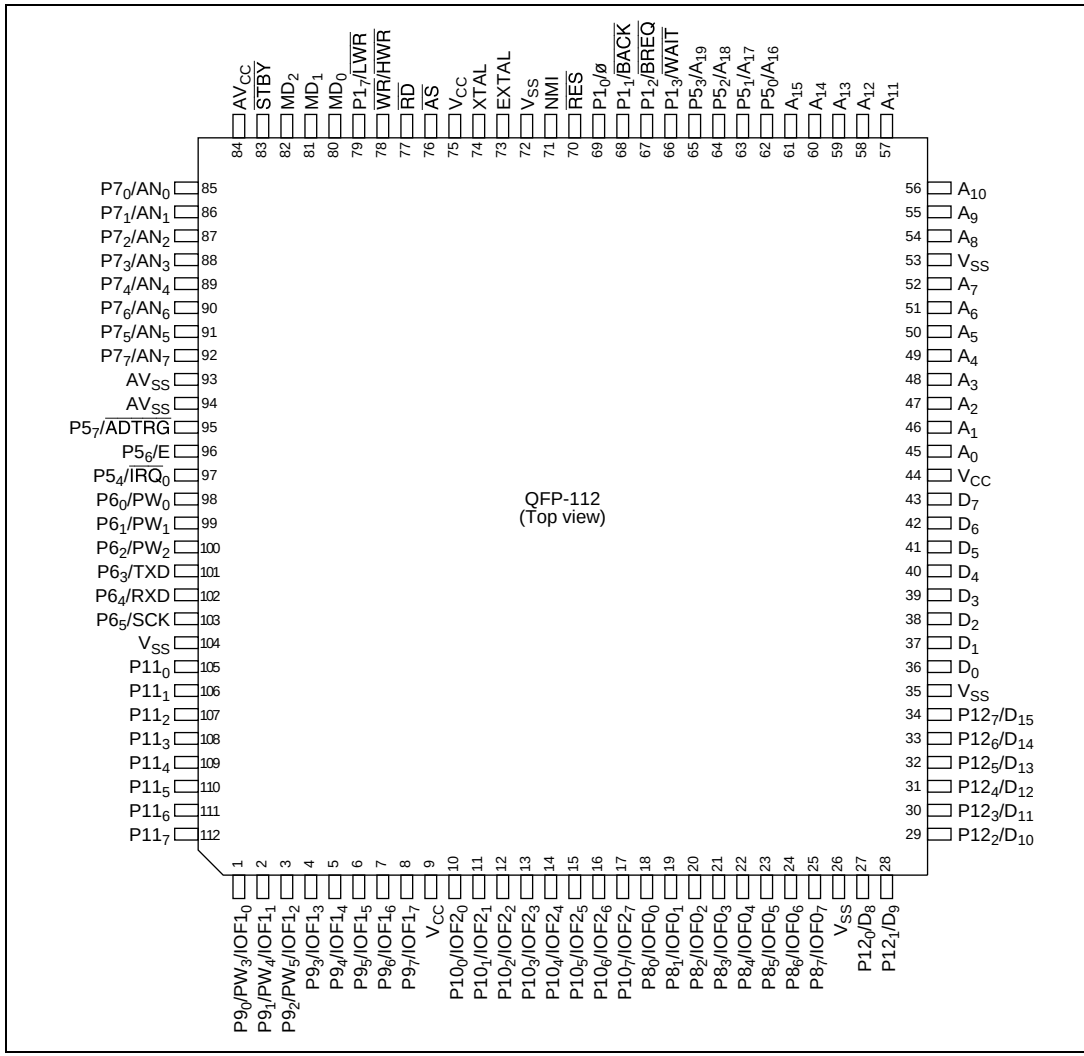


Figure 1.2 H8/570 Pin Arrangement

1.3.2 Pin Functions

Table 1.2 Pin Assignments in Each Operating Mode

	Pin Name				
	Expanded Minimum Mode	Expanded Maximum Mode	Expanded Minimum Mode	Expanded Maximum Modes	
Pin	Mode 1	Mode 3	Mode 4	Mode 5	Mode 6
1	$P9_0/PW_3/IOF1_0$	$P9_0/PW_3/IOF1_0$	$P9_0/PW_3/IOF1_0$	$P9_0/PW_3/IOF1_0$	$P9_0/PW_3/IOF1_0$
2	$P9_1/PW_4/IOF1_1$	$P9_1/PW_4/IOF1_1$	$P9_1/PW_4/IOF1_1$	$P9_1/PW_4/IOF1_1$	$P9_1/PW_4/IOF1_1$
3	$P9_2/PW_5/IOF1_2$	$P9_2/PW_5/IOF1_2$	$P9_2/PW_5/IOF1_2$	$P9_2/PW_5/IOF1_2$	$P9_2/PW_5/IOF1_2$
4	$P9_3/IOF1_3$	$P9_3/IOF1_3$	$P9_3/IOF1_3$	$P9_3/IOF1_3$	$P9_3/IOF1_3$
5	$P9_4/IOF1_4$	$P9_4/IOF1_4$	$P9_4/IOF1_4$	$P9_4/IOF1_4$	$P9_4/IOF1_4$
6	$P9_5/IOF1_5$	$P9_5/IOF1_5$	$P9_5/IOF1_5$	$P9_5/IOF1_5$	$P9_5/IOF1_5$
7	$P9_6/IOF1_6$	$P9_6/IOF1_6$	$P9_6/IOF1_6$	$P9_6/IOF1_6$	$P9_6/IOF1_6$
8	$P9_7/IOF1_7$	$P9_7/IOF1_7$	$P9_7/IOF1_7$	$P9_7/IOF1_7$	$P9_7/IOF1_7$
9	V_{CC}	V_{CC}	V_{CC}	V_{CC}	V_{CC}
10	$P10_0/IOF2_0$	$P10_0/IOF2_0$	$P10_0/IOF2_0$	$P10_0/IOF2_0$	$P10_0/IOF2_0$
11	$P10_1/IOF2_1$	$P10_1/IOF2_1$	$P10_1/IOF2_1$	$P10_1/IOF2_1$	$P10_1/IOF2_1$
12	$P10_2/IOF2_2$	$P10_2/IOF2_2$	$P10_2/IOF2_2$	$P10_2/IOF2_2$	$P10_2/IOF2_2$
13	$P10_3/IOF2_3$	$P10_3/IOF2_3$	$P10_3/IOF2_3$	$P10_3/IOF2_3$	$P10_3/IOF2_3$
14	$P10_4/IOF2_4$	$P10_4/IOF2_4$	$P10_4/IOF2_4$	$P10_4/IOF2_4$	$P10_4/IOF2_4$
15	$P10_5/IOF2_5$	$P10_5/IOF2_5$	$P10_5/IOF2_5$	$P10_5/IOF2_5$	$P10_5/IOF2_5$
16	$P10_6/IOF2_6$	$P10_6/IOF2_6$	$P10_6/IOF2_6$	$P10_6/IOF2_6$	$P10_6/IOF2_6$
17	$P10_7/IOF2_7$	$P10_7/IOF2_7$	$P10_7/IOF2_7$	$P10_7/IOF2_7$	$P10_7/IOF2_7$
18	$P8_0/IOF0_0$	$P8_0/IOF0_0$	$P8_0/IOF0_0$	$P8_0/IOF0_0$	$P8_0/IOF0_0$
19	$P8_1/IOF0_1$	$P8_1/IOF0_1$	$P8_1/IOF0_1$	$P8_1/IOF0_1$	$P8_1/IOF0_1$
20	$P8_2/IOF0_2$	$P8_2/IOF0_2$	$P8_2/IOF0_2$	$P8_2/IOF0_2$	$P8_2/IOF0_2$
21	$P8_3/IOF0_3$	$P8_3/IOF0_3$	$P8_3/IOF0_3$	$P8_3/IOF0_3$	$P8_3/IOF0_3$
22	$P8_4/IOF0_4$	$P8_4/IOF0_4$	$P8_4/IOF0_4$	$P8_4/IOF0_4$	$P8_4/IOF0_4$
23	$P8_5/IOF0_5$	$P8_5/IOF0_5$	$P8_5/IOF0_5$	$P8_5/IOF0_5$	$P8_5/IOF0_5$
24	$P8_6/IOF0_6$	$P8_6/IOF0_6$	$P8_6/IOF0_6$	$P8_6/IOF0_6$	$P8_6/IOF0_6$
25	$P8_7/IOF0_7$	$P8_7/IOF0_7$	$P8_7/IOF0_7$	$P8_7/IOF0_7$	$P8_7/IOF0_7$
26	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
27	D_8	D_8	$P12_0$	D_8	$P12_0$
28	D_9	D_9	$P12_1$	D_9	$P12_1$
29	D_{10}	D_{10}	$P12_2$	D_{10}	$P12_2$

	Pin Name				
	Expanded Minimum Mode	Expanded Maximum Mode	Expanded Minimum Mode	Expanded Maximum Modes	
Pin	Mode 1	Mode 3	Mode 4	Mode 5	Mode 6
30	D_{11}	D_{11}	$P12_3$	D_{11}	$P12_3$
31	D_{12}	D_{12}	$P12_4$	D_{12}	$P12_4$
32	D_{13}	D_{13}	$P12_5$	D_{13}	$P12_5$
33	D_{14}	D_{14}	$P12_6$	D_{14}	$P12_6$
34	D_{15}	D_{15}	$P12_7$	D_{15}	$P12_7$
35	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
36	D_0	D_0	D_0	D_0	D_0
37	D_1	D_1	D_1	D_1	D_1
38	D_2	D_2	D_2	D_2	D_2
39	D_3	D_3	D_3	D_3	D_3
40	D_4	D_4	D_4	D_4	D_4
41	D_5	D_5	D_5	D_5	D_5
42	D_6	D_6	D_6	D_6	D_6
43	D_7	D_7	D_7	D_7	D_7
44	V_{CC}	V_{CC}	V_{CC}	V_{CC}	V_{CC}
45	A_0	A_0	A_0	A_0	A_0
46	A_1	A_1	A_1	A_1	A_1
47	A_2	A_2	A_2	A_2	A_2
48	A_3	A_3	A_3	A_3	A_3
49	A_4	A_4	A_4	A_4	A_4
50	A_5	A_5	A_5	A_5	A_5
51	A_6	A_6	A_6	A_6	A_6
52	A_7	A_7	A_7	A_7	A_7
53	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
54	A_8	A_8	A_8	A_8	A_8
55	A_9	A_9	A_9	A_9	A_9
56	A_{10}	A_{10}	A_{10}	A_{10}	A_{10}
57	A_{11}	A_{11}	A_{11}	A_{11}	A_{11}
58	A_{12}	A_{12}	A_{12}	A_{12}	A_{12}
59	A_{13}	A_{13}	A_{13}	A_{13}	A_{13}
60	A_{14}	A_{14}	A_{14}	A_{14}	A_{14}
61	A_{15}	A_{15}	A_{15}	A_{15}	A_{15}

	Pin Name				
	Expanded Minimum Mode	Expanded Maximum Mode	Expanded Minimum Mode	Expanded Maximum Modes	
Pin	Mode 1	Mode 3	Mode 4	Mode 5	Mode 6
62	$P5_0$	A_{16}	$P5_0$	A_{16}	A_{16}
63	$P5_1$	A_{17}	$P5_1$	A_{17}	A_{17}
64	$P5_2$	A_{18}	$P5_2$	A_{18}	A_{18}
65	$P5_3$	A_{19}	$P5_3$	A_{19}	A_{19}
66	$P1_3/\overline{WAIT}$	$P1_3/\overline{WAIT}$	$P1_3/\overline{WAIT}$	$P1_3/\overline{WAIT}$	$P1_3/\overline{WAIT}$
67	$P1_2/\overline{BREQ}$	$P1_2/\overline{BREQ}$	$P1_2/\overline{BREQ}$	$P1_2/\overline{BREQ}$	$P1_2/\overline{BREQ}$
68	$P1_1/\overline{BACK}$	$P1_1/\overline{BACK}$	$P1_1/\overline{BACK}$	$P1_1/\overline{BACK}$	$P1_1/\overline{BACK}$
69	$P1_0$/ø	$P1_0$/ø	$P1_0$/ø	$P1_0$/ø	$P1_0$/ø
70	$\overline{RES}$	$\overline{RES}$	$\overline{RES}$	$\overline{RES}$	$\overline{RES}$
71	NMI	NMI	NMI	NMI	NMI
72	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
73	EXTAL	EXTAL	EXTAL	EXTAL	EXTAL
74	XTAL	XTAL	XTAL	XTAL	XTAL
75	V_{CC}	V_{CC}	V_{CC}	V_{CC}	V_{CC}
76	$\overline{AS}$	$\overline{AS}$	$\overline{AS}$	$\overline{AS}$	$\overline{AS}$
77	$\overline{RD}$	$\overline{RD}$	$\overline{RD}$	$\overline{RD}$	$\overline{RD}$
78	$\overline{HWR}$	$\overline{HWR}$	$\overline{WR}$	$\overline{HWR}$	$\overline{WR}$
79	$\overline{LWR}$	$\overline{LWR}$	$P1_7$	$\overline{LWR}$	$P1_7$
80	MD_0	MD_0	MD_0	MD_0	MD_0
81	MD_1	MD_1	MD_1	MD_1	MD_1
82	MD_2	MD_2	MD_2	MD_2	MD_2
83	$\overline{STBY}$	$\overline{STBY}$	$\overline{STBY}$	$\overline{STBY}$	$\overline{STBY}$
84	AV_{CC}	AV_{CC}	AV_{CC}	AV_{CC}	AV_{CC}
85	$P7_0/AN_0$	$P7_0/AN_0$	$P7_0/AN_0$	$P7_0/AN_0$	$P7_0/AN_0$
86	$P7_1/AN_1$	$P7_1/AN_1$	$P7_1/AN_1$	$P7_1/AN_1$	$P7_1/AN_1$
87	$P7_2/AN_2$	$P7_2/AN_2$	$P7_2/AN_2$	$P7_2/AN_2$	$P7_2/AN_2$
88	$P7_3/AN_3$	$P7_3/AN_3$	$P7_3/AN_3$	$P7_3/AN_3$	$P7_3/AN_3$
89	$P7_4/AN_4$	$P7_4/AN_4$	$P7_4/AN_4$	$P7_4/AN_4$	$P7_4/AN_4$
90	$P7_5/AN_5$	$P7_5/AN_5$	$P7_5/AN_5$	$P7_5/AN_5$	$P7_5/AN_5$
91	$P7_6/AN_6$	$P7_6/AN_6$	$P7_6/AN_6$	$P7_6/AN_6$	$P7_6/AN_6$
92	$P7_7/AN_7$	$P7_7/AN_7$	$P7_7/AN_7$	$P7_7/AN_7$	$P7_7/AN_7$
93	AV_{SS}	AV_{SS}	AV_{SS}	AV_{SS}	AV_{SS}

	Pin Name				
	Expanded Minimum Mode	Expanded Maximum Mode	Expanded Minimum Mode	Expanded Maximum Modes	
Pin	Mode 1	Mode 3	Mode 4	Mode 5	Mode 6
94	AV_{SS}	AV_{SS}	AV_{SS}	AV_{SS}	AV_{SS}
95	$P5_7/\overline{ADTRG}$	$P5_7/\overline{ADTRG}$	$P5_7/\overline{ADTRG}$	$P5_7/\overline{ADTRG}$	$P5_7/\overline{ADTRG}$
96	$P5_6$/E	$P5_6$/E	$P5_6$/E	$P5_6$/E	$P5_6$/E
97	$P5_4/\overline{IRQ}_0$	$P5_4/\overline{IRQ}_0$	$P5_4/\overline{IRQ}_0$	$P5_4/\overline{IRQ}_0$	$P5_4/\overline{IRQ}_0$
98	$P6_0/PW_0$	$P6_0/PW_0$	$P6_0/PW_0$	$P6_0/PW_0$	$P6_0/PW_0$
99	$P6_1/PW_1$	$P6_1/PW_1$	$P6_1/PW_1$	$P6_1/PW_1$	$P6_1/PW_1$
100	$P6_2/PW_2$	$P6_2/PW_2$	$P6_2/PW_2$	$P6_2/PW_2$	$P6_2/PW_2$
101	$P6_3$/TXD	$P6_3$/TXD	$P6_3$/TXD	$P6_3$/TXD	$P6_3$/TXD
102	$P6_4$/RXD	$P6_4$/RXD	$P6_4$/RXD	$P6_4$/RXD	$P6_4$/RXD
103	$P6_5$/SCK	$P6_5$/SCK	$P6_5$/SCK	$P6_5$/SCK	$P6_5$/SCK
104	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
105	$P11_0$	$P11_0$	$P11_0$	$P11_0$	$P11_0$
106	$P11_1$	$P11_1$	$P11_1$	$P11_1$	$P11_1$
107	$P11_2$	$P11_2$	$P11_2$	$P11_2$	$P11_2$
108	$P11_3$	$P11_3$	$P11_3$	$P11_3$	$P11_3$
109	$P11_4$	$P11_4$	$P11_4$	$P11_4$	$P11_4$
110	$P11_5$	$P11_5$	$P11_5$	$P11_5$	$P11_5$
111	$P11_6$	$P11_6$	$P11_6$	$P11_6$	$P11_6$
112	$P11_7$	$P11_7$	$P11_7$	$P11_7$	$P11_7$

Table 1.3 Pin Functions

Type	Symbol	Pin	I/O	Name and Function
Power	V_{CC}	9, 44, 75	I	**Power:** Connect the system power supply (+5 V) to these pins. The chip will not operate if any V_{CC} pin is left unconnected.
	V_{SS}	26, 35, 53, 72, 104	I	**Ground:** Connect the system ground (0 V) to these pins. The chip will not operate if any V_{SS} pin is left unconnected.
Clock	XTAL	74	I	**Crystal:** Connect a crystal oscillator to this pin. The crystal frequency should be double the desired ø clock frequency. If an external clock is input at the EXTAL pin, leave the XTAL pin unconnected.
	EXTAL	73	I	**External crystal:** Connect a crystal oscillator or external clock to this pin. The frequency of the external clock should be double the desired ø clock frequency. See section 8.2, Oscillator Circuit, for examples of connections to a crystal or external clock.
	ø	69	O	**System clock:** This pin supplies the ø clock to peripheral devices.
	E	96	O	**Enable clock:** This pin supplies an E clock to E-clock-based peripheral devices.
System control	$\overline{\text{BACK}}$	68	O	**Bus request acknowledge:** This signal indicates that the bus access right has been granted to an external device. $\overline{\text{BACK}}$ notifies the external device that issued a $\overline{\text{BREQ}}$ signal that it now has control of the bus.
	$\overline{\text{BREQ}}$	67	I	**Bus request:** This signal is sent by an external device to the H8/570 MCU to request the bus access right.
	$\overline{\text{STBY}}$	83	I	**Standby:** A transition to the hardware standby mode (a power-down state) occurs when this pin is driven low.
	$\overline{\text{RES}}$	70	I/O	**Reset:** Driving $\overline{\text{RES}}$ low causes the H8/570 MCU to reset. Overflow of WDT outputs a reset request signal.
Address bus	A_{19}–A_0	65–54, 52–45	O	**Address bus:** Address output pins.
Data bus	D_7–D_0 D_8–D_{15}	43–36 34–27	I/O	**Data bus:** Bidirectional data bus.

Type	Symbol	Pin	I/O	Name and Function
Bus control	$\overline{WAIT}$	66	I	**Wait:** This signal requests the CPU to insert one or more T_w states when accessing an external address.
	$\overline{AS}$	76	O	**Address strobe:** A low output signal from this pin indicates that there is a valid address on the address bus.
	$\overline{RD}$	77	O	**Read:** A low output signal from this pin indicates that the CPU is reading an external address.
	$\overline{WR}$	78	O	**Write:** A low output signal from this pin indicates that the CPU is writing to an external address.
	$\overline{HWR}$	78	O	**Upper byte write:** A low output signal from this pin indicates that the CPU is writing to the upper byte of an external address.
	$\overline{LWR}$	79	O	**Lower byte write:** A low output signal from this pin indicates that the CPU is writing to the lower byte of an external address.
Interrupt	NMI	71	I	**Nonmaskable interrupt:** Highest priority interrupt request. System control register 1 (SYSCR1) determines whether the interrupt is requested on the rising or falling edge of the NMI input.
	$\overline{IRQ_0}$	97	I	**Interrupt request 0:** Maskable interrupt request pins.
Operating mode control	MD_2 MD_1 MD_0	82 81 80	I	**Mode:** These input pins are used for setting the MCU operating mode according to the table below.

MD_2	MD_1	MD_0	Mode	Description
0	0	0	Mode 0	—
		1	Mode 1	16-bit expanded minimum mode
	1	0	Mode 2	Reserved
		1	Mode 3	16-bit expanded maximum mode
1	0	0	Mode 4	8-bit expanded minimum mode
		1	Mode 5	16-bit expanded maximum mode
	1	0	Mode 6	8-bit expanded maximum mode
		1	Mode 7	Reserved

Do not change values MD_2 to MD_0 during the MCU operation. If these values are changed during the operation, the H8/570 MCU will not operate correctly. The inputs at these pins are latched in mode select bits 2 to 0 (MDS2–MDS0) of the mode control register (MDCR).

Type	Symbol	Pin	I/O	Name and Function
PWM timer	PW_0 PW_1 PW_2 PW_3 PW_4 PW_5	98 99 100 1 2 3	O	**PWM timer output:** Pulse-width-modulation timer output pulses.
Serial communication interface signals	TXD	101	O	**Transmit data:** Data output pins for the serial communication interface.
	RXD	102	I	**Receive data:** Data input pins for the serial communication interface.
	SCK	103	I/O	**Serial clock:** Input/output pin for the serial interface clock.
A/D converter	AN_7–AN_0	92–85	I	**Analog input:** Analog signal input pins.
	AV_{CC}	84	I	**Analog reference voltage:** Reference voltage and power supply pin for the A/D converter.
	AV_{SS}	93, 94	I	**Analog ground:** Ground pin for the A/D converter.
	$\overline{ADTRG}$	95	I	**External trigger input:** External trigger input pins for the A/D converter.
Parallel I/O	$P1_7$ $P1_3$–$P1_0$	79 69–66	I/O	**Port 1:** A 5-bit input/output port. The direction of each bit is determined by the port 1 data direction register (P1DDR).
	$P5_7$, $P5_6$ $P5_4$	95, 96 97	I/O	**Port 5:** A 7-bit input/output port. The direction of each bit is determined by the port 5 data direction register (P5DDR).
	$P5_3$–$P5_0$	65–62	I/O	Pins $P5_3$ to $P5_0$ have built-in MOS input pull-ups.
	$P6_5$–$P6_0$	103–98	I/O	**Port 6:** A 6-bit input/output port. The direction of each bit is determined by the port 6 data direction register (P6DDR).
	$P7_7$–$P7_0$	92–85	I	**Port 7:** An 8-bit input port.
	$P8_7$–$P8_0$	25–18	I/O	**Port 8:** An 8-bit input/output port. The direction of each bit is determined by the port 8 data direction register (P8DDR).
	$P9_7$–$P9_0$	8–1	I/O	**Port 9:** An 8-bit input/output port. The direction of each bit is determined by the port 9 data direction register (P9DDR).
	$P10_7$– $P10_0$	17–10	I/O	**Port 10:** An 8-bit input/output port. The direction of each bit is determined by the port 10 data direction register (P10DDR).

Type	Symbol	Pin	I/O	Name and Function
Parallel I/O	$P11_7$–$P11_0$	112–105	I/O	**Port 11:** An 8-bit input/output port. The direction of each bit is determined by the port 11 data direction register (P11DDR).
	$P12_7$–$P12_0$	34–27	I/O	**Port 12:** An 8-bit input/output port. The direction of each bit is determined by the port 12 data direction register (P12DDR).
ISP pins	$IOF0_7$–$IOF0_0$	25–18	I/O	**ISP I/O flag 0:** Input/output pins for the ISP module.
	$IOF1_7$–$IOF1_0$	8–1	I/O	**ISP I/O flag 1:** Input/output pins for the ISP module.
	$IOF2_7$–$IOF2_0$	17–10	I/O	**ISP I/O flag 2:** Input/output pins for the ISP module.

Section 2 MCU Operating Modes and Address Space

2.1 Overview

The H8/570 microcomputer unit (MCU) operates in five modes numbered 1, 3, 4, 5, and 6. The mode is selected by the inputs at the mode pins (MD_2 to MD_0). The MCU mode (table 2.1) determines the size of the address space, the usage of its RAM, and the operating mode of the CPU. The MCU mode also affects the functions of the I/O pins.

Table 2.1 Operating Modes

MD_2	MD_1	MD_0	MCU Mode	Address Space	RAM	CPU Mode	Data Bus
0	0	0	—	—	—	—	—
		1	Mode 1	Expanded minimum	Enabled*[1]	Minimum mode	16-bit
	1	0	—	—	—	—	—
		1	Mode 3*[2]	Expanded maximum	Enabled*[1]	Maximum mode	16-bit
1	0	0	Mode 4	Expanded minimum	Enabled*[1]	Minimum mode	8-bit
		1	Mode 5*[2]	Expanded maximum	Enabled*[1]	Maximum mode	16-bit
	1	0	Mode 6	Expanded maximum	Enabled*[1]	Maximum mode	8-bit
		1	—	—	—	—	—

Notes: 1. By clearing the RAM enable bit (RAME) in the RAM control register (RAMCR), an external address can be used.
2. For differences between mode 3 and mode 5, refer to section 2.3, Address Map.
—: Indicates unusable

The expanded minimum modes (modes 1 and 4) support a maximum address space of 64 kbytes. The expanded maximum modes (modes 3, 5, and 6) support a maximum address space of 1 Mbyte.

The H8/570 cannot be set to modes 0, 2, and 7. The mode pins should never be set to these values.

2.2 Mode Descriptions

The five MCU modes are described below. For further information on the I/O pin functions in each mode, see section 9, I/O Ports.

Mode 1 (16-Bit Expanded Minimum Mode): Mode 1 supports a maximum of 64 kbytes of address space. Port 12 is used for upper data bus D_{15} to D_8.

Mode 3 (16-Bit Expanded Maximum Mode): Mode 3 supports a maximum of 1 Mbyte of address space. Port 12 is used for upper data bus D_{15} to D_8, and $P5_3$ to $P5_0$ of port 5 are used for address bus A_{19} to A_{16}. Before using port 5 for the address bus, the software must change the corresponding P5DDR to outputs. See section 9.3, Port 5, for details. As a note, in mode 3, page 15 is fixed in byte-wide space and includes the address for internal I/O.

Mode 4 (8-Bit Expanded Minimum Mode): Mode 4 supports a maximum of 64 kbytes of address space.

Mode 5 (16-Bit Expanded Maximum Mode): Mode 5 supports a maximum of 1 Mbyte of address space. Port 12 is used for upper data bus D_{15} to D_8, and $P5_3$ to $P5_0$ of port 5 are used for address bus A_{19} to A_{16}.

Mode 6 (8-Bit Expanded Maximum Mode): Mode 6 supports a maximum of 1 Mbyte of address space. $P5_3$ to $P5_0$ of port 5 are used for address bus A_{19} to A_{16}. As a note, in mode 6, page 15 includes the address for internal I/O.

2.3 Address Map

The H8/570's address space is segmented into 64-kbyte pages. In the expanded minimum modes there is just one page: page 0. In the expanded maximum modes there can be up to 16 pages. The address space for each mode are made up of on-chip and external segments.

In mode 1, addresses for on-chip registers other than those in the ISP and addresses ranging from H'FF80 to H'FFFF are assigned to the byte-wide space.

In mode 3, addresses for on-chip registers other than those in the ISP, addresses ranging from H'0FF80 to H'FFFFF, and page 15 are assigned to byte-wide space.

In mode 5, addresses for on-chip registers other than those in the ISP and addresses ranging from H'0FF80 to H'FFFFF are assigned to byte-wide space.

In modes 4 and 6, addresses for the on-chip RAM and registers in the ISP are assigned to word-wide space.

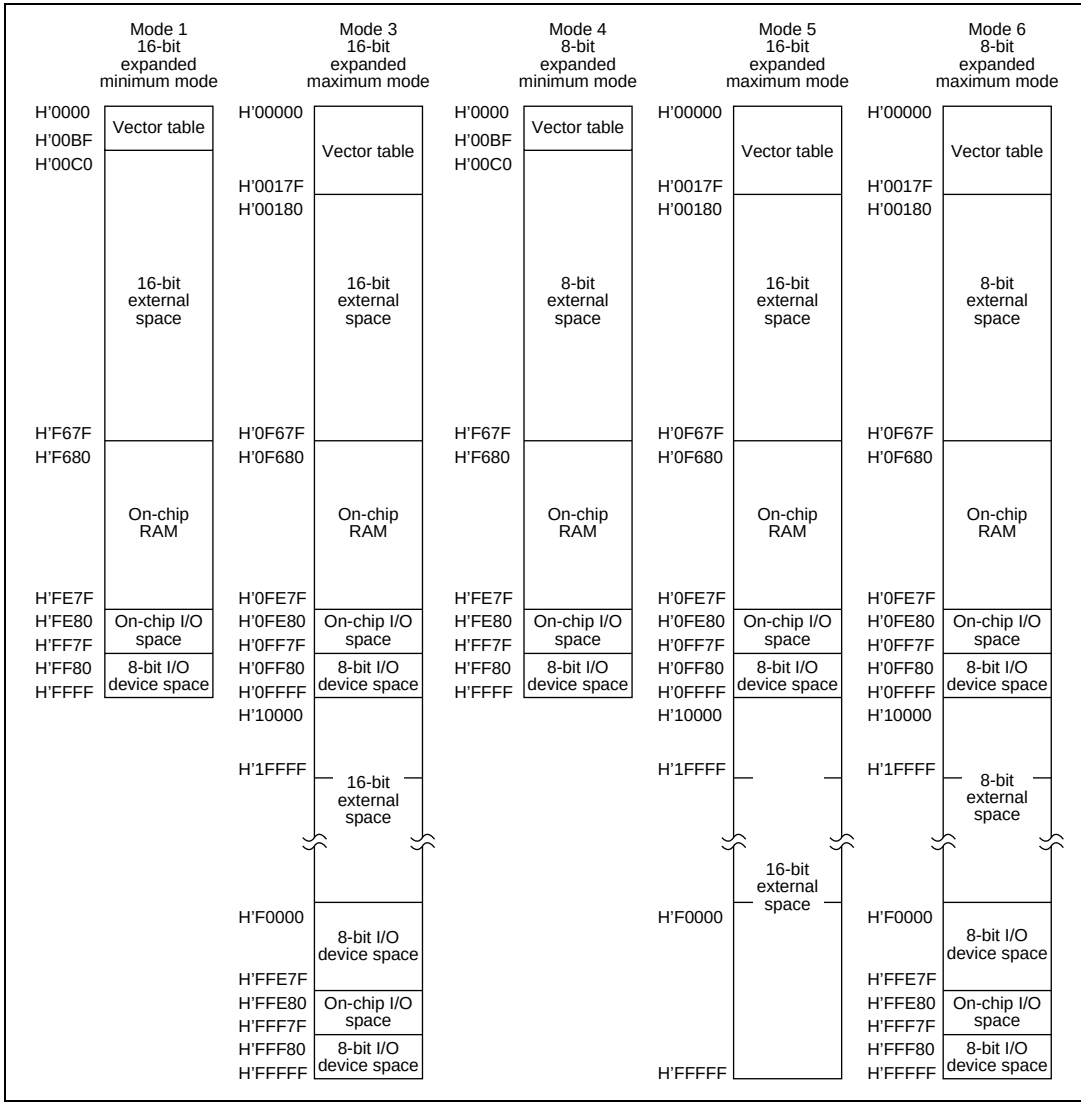


Figure 2.1 Address Space in Each Mode

2.4 Mode Control Register (MDCR)

The mode control register (MDCR) is a control register located in the register field in page 0. The inputs at the mode pins are latched to this register. The mode control register can only be read by the CPU.

Name	Abbreviation	Read/Write	Address
Mode control register	MDCR	Read only	H'FF4A

The bit configuration of this register is shown below.

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	—
Initial value	1	1	0	0	0	*	*	*
Read/Write	—	—	—	—	—	R	R	R

Note: * Initialized according to MD_2 to MD_0.

Bits 7 and 6—Reserved: These bits cannot be modified and are always read as 1.

Bits 5 to 3—Reserved: These bits cannot be modified and are always read as 0.

Bits 2 to 0—Mode Select 2 to 0 (MDS2 to MDS0): These bits indicate the values of the mode pins (MD_2 to MD_0). MDS2, MDS1, and MDS0 correspond to MD2, MD1, and MD0, respectively. These bits are read-only bits.

Section 3 CPU

3.1 Overview

The H8/570 MCU contains the H8/500 Series CPU: a high-speed central processing unit designed for realtime control of a wide range of medium-scale office and industrial equipment. Its Hitachi-original architecture features eight 16-bit general registers, internal 16-bit data paths, and an optimized instruction set.

This section summarizes the CPU architecture and its instruction set.

3.1.1 Features

- General-register machine
  - Eight 16-bit general registers
  - Seven control registers (two 16-bit registers, five 8-bit registers)
- High speed: Maximum at 12 MHz (a register-register add operation takes only 167 ns)
- Address space managed in 64-kbyte pages, expandable to 1 Mbyte*
  - Page registers: Simultaneous four-page availability (code page, stack page, data page, and extended page)
- Two CPU operating modes:
  - Minimum mode: Maximum 64-kbyte address space
  - Maximum mode: Maximum 1-Mbyte address space*
- Highly-orthogonal instruction set
  - Addressing modes and data sizes can be specified independently within each instruction
- 1.5 addressing modes
  - Register-register and register-memory operations are supported
- Optimized for efficient programming in C language
  - Special short formats for frequently-used instructions and addressing modes

Note: * The CPU architecture will support up to 16 Mbytes of external memory for future versions.

3.1.2 Address Space

The size of the address space depends on the operating mode.

The H8/570 MCU has five operating modes, which are selected by the input to the mode pins (MD_2 to MD_0) after the MCU has been reset. The CPU, however, has only two operating modes. The MCU operating mode determines the CPU operating mode, which in turn determines the maximum address space size (figure 3.1).

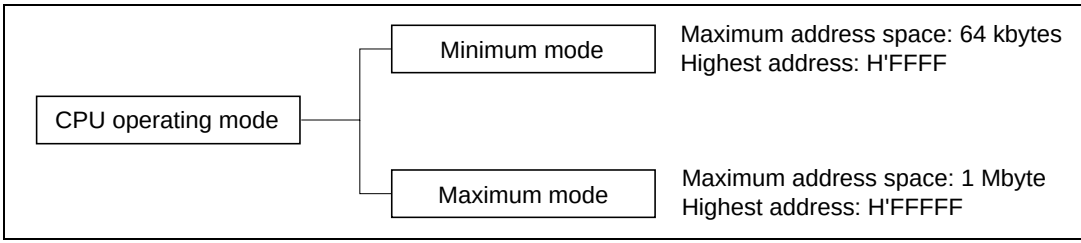


Figure 3.1 CPU Operating Modes

3.1.3 Register Configuration

The register structure of the CPU are composed of two groups of registers: the general registers (Rn) and control registers (CR).

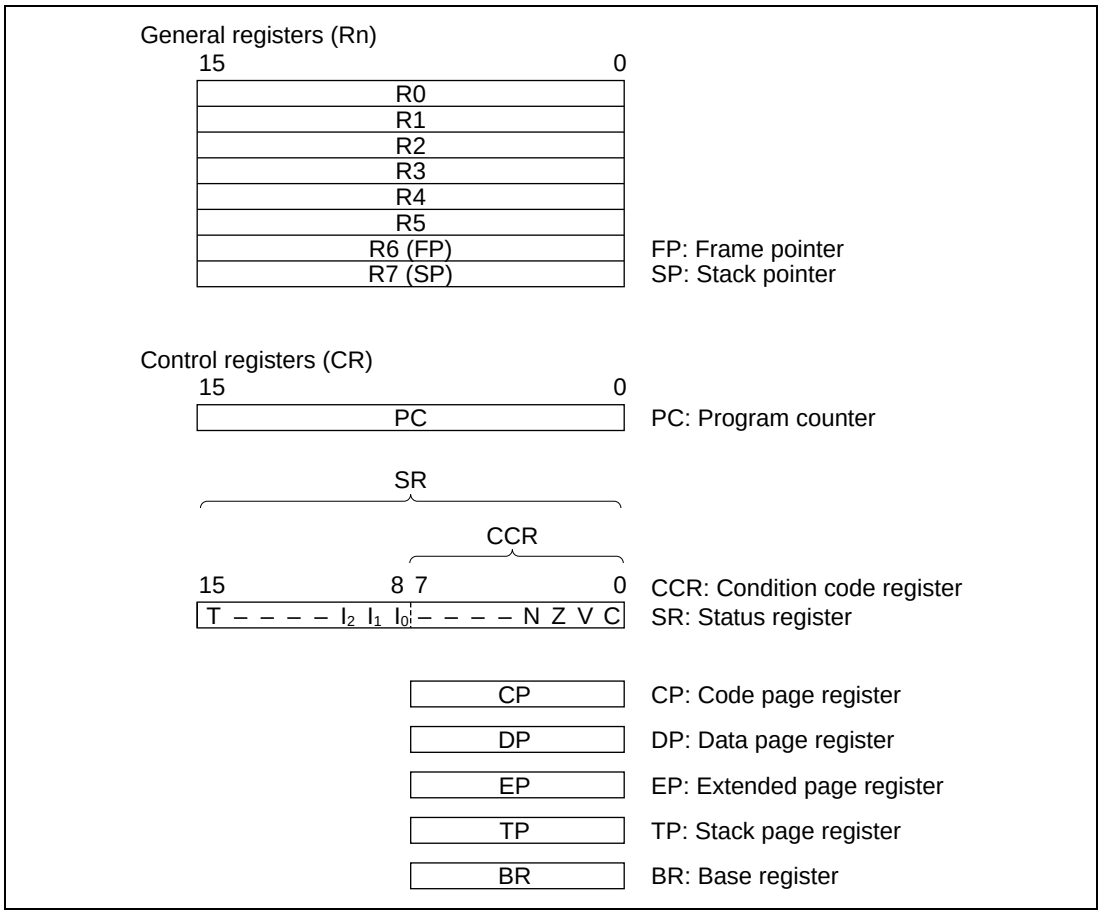


Figure 3.2 Registers in the CPU

3.2 CPU Register Descriptions

3.2.1 General Registers

All eight of the 16-bit general registers are functionally alike; there is no distinction between data registers and address registers. When these registers are accessed as data registers, either byte or word size can be selected.

R6 and R7, in addition to functioning as general registers, have special assignments.

R7 is the stack pointer, used implicitly in exception handling and subroutine calls. It can be designated as SP, which is synonymous with R7. The SP always points to the top of the stack (figure 3.3). The SP is also used implicitly by the LDM and STM instructions, which load and store multiple registers from and to the stack, respectively, and then followed by a pre-decrement or post-increment of R7 accordingly.

R6 functions as a frame pointer (FP). The LINK and UNLK use R6 implicitly to reserve or release a stack frame, respectively.

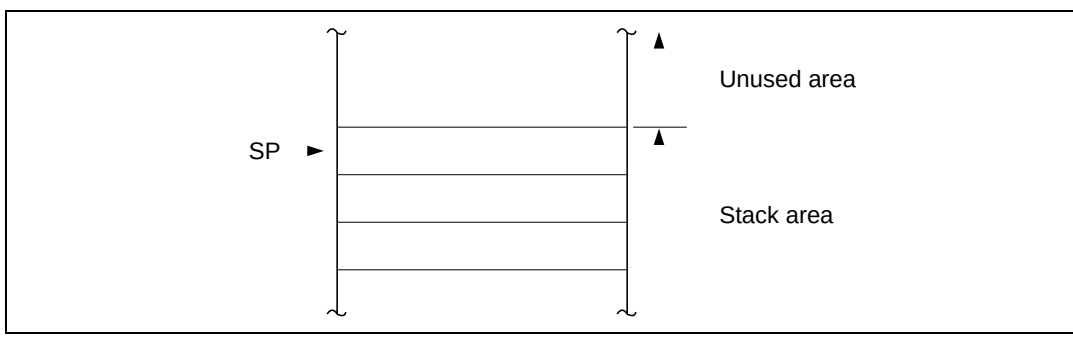


Figure 3.3 Stack Pointer

3.2.2 Control Registers

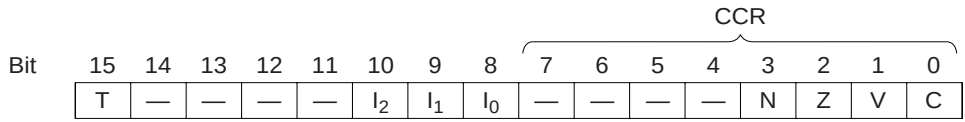
The CPU control registers (CR) include a 16-bit program counter (PC), a 16-bit status register (SR), four 8-bit page registers, and one 8-bit base register (BR).

Program Counter (PC)

This 16-bit register indicates the address of the next instruction the CPU will execute.

Status Register (SR)

This 16-bit register contains internal status information. The lower half of the status register is referred to as the condition code register (CCR): it can be accessed as a separate condition code byte.



Bit 15—Trace (T): When this bit is set to 1, the CPU operates in trace mode and generates a trace exception after every instruction. See section 4.4, Trace, for a description of the trace exception-handling sequence.

When the value of this bit is 0, instructions are executed in a normal continuous sequence. This bit is cleared to 0 at reset.

Bits 14 to 11—Reserved: These bits cannot be modified and are always read as 0.

Bits 10 to 8—Interrupt Mask (I_2, I_1, I_0): These bits indicate the interrupt request mask level (0 to 7). An interrupt request is not accepted unless it has a higher level than the present value of the mask (table 3.1). A nonmaskable interrupt (NMI), which has a level of 8, is accepted at any mask level. After an interrupt is accepted, I_2, I_1, and I_0 are changed to the level of the requested interrupt. Table 3.2 indicates the values of the I bits after an interrupt is accepted.

A reset sets all three bits (I_2, I_1, and I_0) to 1, masking all interrupts except NMI.

Table 3.1 Interrupt Mask Levels

Priority	Mask Level	Mask Bits I_2	I_1	I_0	Interrupts Accepted
High	7	1	1	1	NMI
↑	6	1	1	0	Level 7 and NMI
	5	1	0	1	Levels 6 to 7 and NMI
	4	1	0	0	Levels 5 to 7 and NMI
	3	0	1	1	Levels 4 to 7 and NMI
	2	0	1	0	Levels 3 to 7 and NMI
	1	0	0	1	Levels 2 to 7 and NMI
Low	0	0	0	0	Levels 1 to 7 and NMI

Table 3.2 Interrupt Mask Bits After an Interrupt is Accepted

Level of Interrupt Accepted	Mask Bits I_2	I_1	I_0
NMI (8)	1	1	1
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1

Bits 7 to 4—Reserved: These bits cannot be modified and are always read as 0.

Bit 3—Negative (N): This bit indicates the most significant bit (sign bit) of the result of an instruction.

Bit 2—Zero (Z): This bit is set to 1 to indicate a zero result and cleared to 0 to indicate a nonzero result.

Bit 1—Overflow (V): This bit is set to 1 when an arithmetic overflow occurs, otherwise, it is cleared to 0.

Bit 0—Carry (C): This bit is set to 1 when a carry or borrow occurs at the most significant bit, otherwise, it is cleared to 0 or left unchanged.

The specific changes that occur in the condition code bits when each instruction is executed are listed in appendix A.1, Instruction Set. See the *H8/500 Series Programming Manual* for further details.

Page Registers

The code page register (CP), data page register (DP), extended page register (EP), and stack page register (TP) are 8-bit registers that are used only in the maximum mode. Their contents are not used in the minimum mode.

In the maximum mode, the page registers are combined with the program counter and general registers to generate 24-bit effective addresses (figure 3.4), thereby expanding the program area, data area, and stack area.

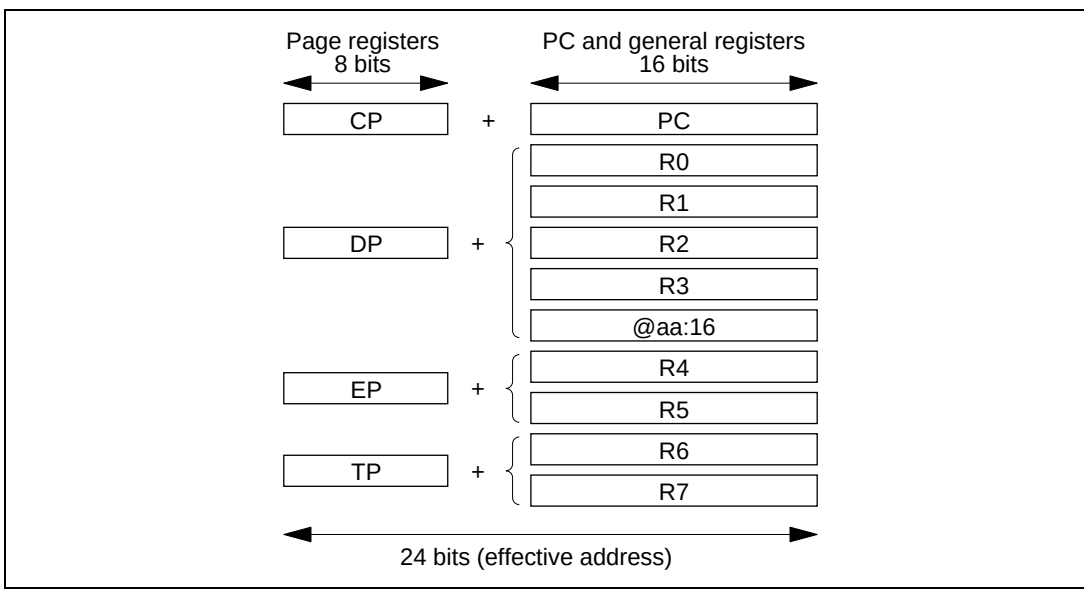


Figure 3.4 Combinations of Page Registers with Other Registers

Code Page Register (CP): The code page register and the program counter combine to generate a 24-bit program code address. In the maximum mode, the code page register is initialized at reset and loaded with a value from the vector table, and both the code page register and program counter are saved and restored during exception handling.

Data Page Register (DP): The data page register combines with one of the general registers R0 to R3 to generate a 24-bit effective address. The data page register contains the upper 8 bits of the address. It is used to calculate effective addresses in the register indirect addressing mode using R0 to R3 and in the 16-bit absolute addressing mode (@aa:16).

The data page register is rewritten by the LDC instruction.

Extended Page Register (EP): The extended page register combines with the general register R4 or R5 to generate a 24-bit operand address. The extended page register contains the upper 8 bits of the address. It is used to calculate effective addresses in the register indirect addressing mode using R4 or R5.

The extended page can be used as an additional data page.

Stack Page Register (TP): The stack page register combines with R6 (FP) or R7 (SP) to generate a 24-bit stack address. The stack page register contains the upper 8 bits of the address. It is used to calculate effective addresses in the register indirect addressing mode using R6 or R7, in exception handling and for subroutine calls.

Base Register (BR)

This 8-bit register stores the base address used in the short absolute addressing mode (@aa:8). In this addressing mode a 16-bit effective address in page 0 is generated by using the contents of the base register as the upper 8 bits and an address given in the instruction code as the lower 8 bits. See figure 3.5.

In the short absolute addressing mode the address is always located in page 0.

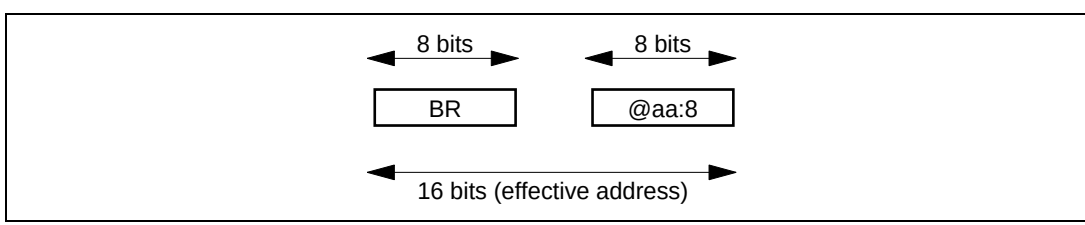


Figure 3.5 Short Absolute Addressing Mode and Base Register

3.2.3 Initial Register Values

When the CPU is reset, its internal registers are initialized (table 3.3). Note that the stack pointer (R7) and base register (BR) are not initialized to fixed values. Also, of the page registers used in maximum mode, only the code page register (CP) is initialized; the other three page registers become undetermined values after being reset.

Accordingly, in the minimum mode the first instruction executed after reset should initialize the stack pointer. The base register must also be initialized before the short absolute addressing mode (@aa:8) is used.

In the maximum mode, the first instruction executed after reset should initialize the stack page register (TP) and the next instruction should initialize the stack pointer. Following instructions should initialize the base register and the other page registers as necessary.

Table 3.3 Initial Values of Registers

Register	Initial Value	
	Minimum Mode	Maximum Mode
General registers 15 0 R7–R0	Undetermined	Undetermined
Control registers 15 0 PC	Loaded from vector table	Loaded from vector table
SR CCR 15 8 7 0 T ———— I_2 I_1 I_0 ———— N Z V C	H'070x (x: Undetermined)	H'070x (x: Undetermined)
7 0 CP	Undetermined	Undetermined
7 0 DP	Undetermined	Undetermined
7 0 EP	Undetermined	Undetermined
7 0 TP	Undetermined	Undetermined
7 0 BR	Undetermined	Undetermined

3.3 Data Formats

The H8/500 CPU can process 1-bit data, 4-bit BCD data, 8-bit (byte) data, 16-bit (word) data, and 32-bit (longword) data.

- Bit manipulation instructions operate on 1-bit data
- Decimal arithmetic instructions operate on 4-bit BCD data
- Almost all instructions operate on byte and word data
- Multiply and divide instructions operate on longword data

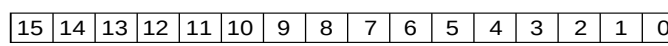
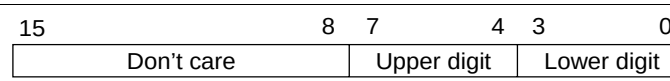
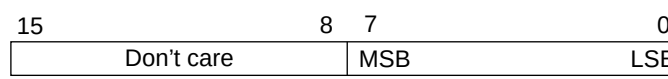
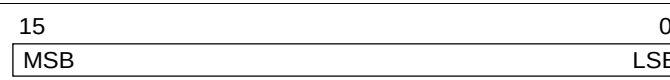
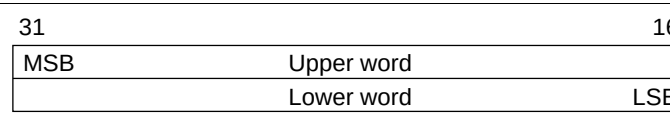
3.3.1 Data Formats in General Registers

Data of any size can be stored in general registers (table 3.4).

Bit data locations are specified by bit number with bit 15 as the most significant bit and bit 0 as the least significant bit. BCD and byte data are stored in the lower 8 bits of a general register. Word data use all 16 bits of a general register. Longword data use two general registers: the upper 16 bits are stored in Rn (n being an even number); the lower 16 bits are stored in Rn + 1.

Operations performed with BCD or byte data do not affect the upper 8 bits of the register.

Table 3.4 General Register Data Formats

Data Type	Register No.	Data Structure
1 bit	Rn	
BCD	Rn	
Byte	Rn	
Word	Rn	
Longword	Rn*	
	Rn + 1*	

Note: * For longword data, n must be zero or even (0, 2, 4, or 6).

3.3.2 Data Formats in Memory

Instructions that access bit data in memory have byte or word operands. The instruction specifies a bit number to indicate the specific bit in the operand.

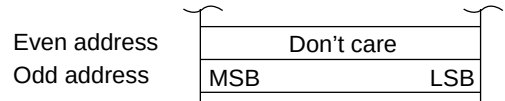
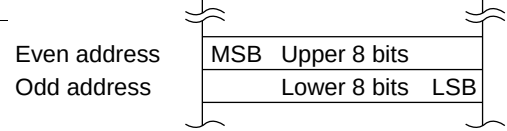
Accessing word data in memory must always begin at an even address. Accessing word data starting at an odd address causes an address error. The upper 8 bits of word data are stored in address n (n being an even number); the lower 8 bits are stored in address n + 1. See table 3.5.

Table 3.5 Data Formats in Memory

Data Type	Data Format	
1 bit (When the operand data is a byte)	Address n	7 6 5 4 3 2 1 0
1 bit (When the operand data is a word)	Even address Odd address	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Byte	Address n	MSB LSB
Word	Even address Odd address	MSB Upper 8 bits Lower 8 bits LSB

When the stack is accessed during exception handling (to save or restore the program counter, code page register, or status register), word-size access is always performed, regardless of the actual data size. Similarly, when the stack is accessed by an instruction using the pre-decrement or post-increment register indirect addressing mode specifying R7 (@–R7 or @R7+), the stack pointer, word-size access is performed regardless of the operand size specified by the instruction. An address error will therefore occur if the stack pointer indicates an odd address. Programs should be coded so that the stack pointer always indicates an even address. See table 3.6.

Table 3.6 Data Formats on the Stack

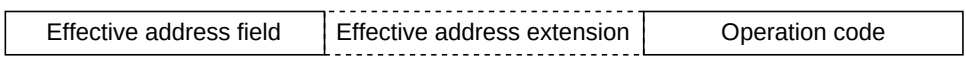
Data Type	Data Format
Byte data on stack	
Word data on stack	

3.4 Instructions

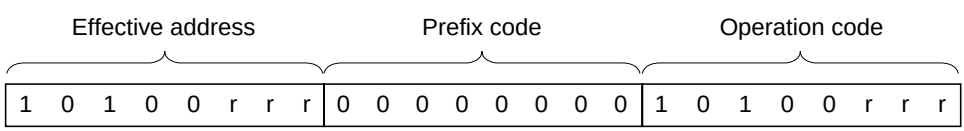
3.4.1 Basic Instruction Formats

There are two basic CPU instruction formats: General format and special format.

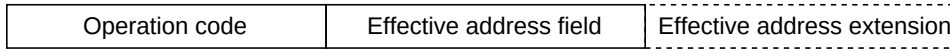
General Format: This format consists of an effective address (EA) field, an effective address extension field, and an operation code (OP) field. The effective address is placed before the operation code for faster instruction execution.



- Effective address field: A one-byte field containing information used to calculate the effective address of an operand.
- Effective address extension: A zero- to two-byte field containing a displacement value, immediate data, or an absolute address. The size of the effective address extension is specified in the effective address field.
- Operation code: This field defines the operation to be carried out on the operand located at the address calculated from the effective address information. Some instructions (DADD, DSUB, MOVFPE, MOVTPE) have an extended format in which the operand code is preceded by a one-byte prefix code.
- Example of the prefix code for the DADD instruction



Special Format: In this format the operation code comes first, followed by the effective address field and effective address extension. This format is used for branching instructions, system control instructions, and other instructions that can be executed faster if the operation is specified before the operand.



- Operation code: A one- or two-byte field defining the operation to be performed by the instruction.
- Effective address field and effective address extension: A zero- to three-byte field containing information used to calculate an effective address.

3.4.2 Addressing Modes

The CPU supports 7 addressing modes: register direct, register indirect, register indirect with displacement, register indirect with pre-decrement or post-increment, immediate, absolute, and PC-relative.

Due to the highly orthogonal nature of the instruction set, most instructions having operands can use any applicable addressing mode except for PC-relative. The PC-relative mode is used by branching instructions.

For most instructions, the addressing mode (table 3.7) is specified in the effective address field. The effective-address extension, if present, contains the displacement, immediate data, or absolute address.

Effective address calculations (table 3.8) are performed for register indirect with displacement, pre-decrement, or post-increment, and for PC-relative.

Table 3.7 Addressing Modes

Addressing Mode	Mnemonic	EA Field	EA Extension
Register direct	Rn	1 0 1 0 Sz r r r	None
Register indirect	@Rn	1 1 0 1 Sz r r r	None
Register indirect with displacement	@(d:8,Rn)	1 1 1 0 Sz r r r	Displacement (1 byte)
	@(d:16,Rn)	1 1 1 1 Sz r r r	Displacement (2 bytes)
Register indirect with:			
pre-decrement	@–Rn	1 0 1 1 Sz r r r	None
post-increment	@Rn+	1 1 0 0 Sz r r r	None
Immediate	#xx:8	0 0 0 0 0 1 0 0	Immediate data (1 byte)
	#xx:16	0 0 0 0 1 1 0 0	Immediate data (2 bytes)
Absolute*	@aa:8	0 0 0 0 Sz 1 0 1	1-byte absolute address (offset from BR)
	@aa:16	0 0 0 1 Sz 1 0 1	2-byte absolute address
PC-relative	disp	No EA field. Addressing mode is specified in the operation code.	1- or 2-byte displacement

Notes: Sz: Specifies the operand size (0 = byte operand, 1 = word operand).

r r r: Register number field, specifying a general register number.

0 0 0: R0
0 0 1: R1
0 1 0: R2
0 1 1: R3
1 0 0: R4
1 0 1: R5
1 1 0: R6
1 1 1: R7

* The @aa:8 addressing mode is also referred to as the short absolute addressing mode.

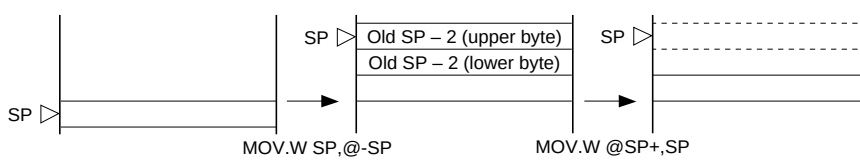
Table 3.8 Effective Address Calculation

Addressing Mode	Effective Address Calculation	Effective Address
Register direct: Rn 1 0 1 0 Sz r r r	—	Operand is the contents of Rn
Register indirect: @Rn 1 1 0 1 Sz r r r	—	23 15 0 DP*1 Rn Also, TP or EP*2
Register indirect with displacement: 8 bits: @(d:8,Rn) 1 1 1 0 Sz r r r	15 0 Rn 15 0 Displacement with sign extension	23 15 0 DP*1 Result Also, TP or EP*2
16 bits: @(d:16,Rn) 1 1 1 1 Sz r r r	15 0 Rn 15 0 Displacement	23 15 0 DP*1 Result Also, TP or EP*2
Register indirect with pre-decrement: @–Rn 1 0 1 1 Sz r r r	15 0 Rn 1 or 2 Rn is decremented by –1 or –2 before instruction execution*3,*4,*5.	23 15 0 DP*1 Result Also, TP or EP*2
Register indirect with post-increment: @Rn+ 1 1 0 0 Sz r r r	Rn is incremented by +1 or +2 after instruction execution*3,*4,*5.	23 15 0 DP*1 Rn Also, TP or EP*2

Addressing Mode	Effective Address Calculation	Effective Address
Absolute address:		
@aa:8 0 0 0 0 Sz 1 0 1	—	23 15 0 H'00 BR EA extension data
@aa:16 0 0 0 1 Sz 1 0 1	—	23 15 0 DP*1 EA extension data
Immediate:		
#xx:8 0 0 0 0 0 1 0 0	—	Operand is the 1-byte EA extension data
#xx:16 0 0 0 0 1 1 0 0	—	Operand is the 2-byte EA extension data
PC-relative;		
8 bits: disp:8, No EA code, Specified in OP code	15 0 PC 15 0 Displacement with sign extension	23 15 0 CP*1 Result
16 bits: disp:16, No EA code, Specified in OP code	15 0 PC 15 0 Displacement	23 15 0 CP*1 Result

Notes: 1. The page register is ignored in minimum mode.

2. The page register used for register indirect depends on the general register: DP for R0, R1, R2, or R3; EP for R4 or R5; TP for R6 or R7.
3. Decrement by –1 for a byte operand, and by –2 for a word operand.
4. The pre-decrement or post-increment is always ±2 when R7 is specified, even if the operand is byte size.
5. The drawing below shows what happens when the @–SP and @ SP+ addressing modes are used to save and restore the stack pointer.



3.5 Instruction Set

3.5.1 Overview

The main features of the CPU instruction set are:

- General-register architecture
- Orthogonality: Addressing modes and data sizes can be specified independently in each instruction
- 1.5 addressing modes (supporting register-register and register-memory operations)
- Ideal for high-level languages, particularly C, with short formats for frequently-used instructions and addressing modes

The CPU instruction set includes 63 types of instructions (table 3.9).

Table 3.9 Instruction Classification

Function (Number of Types)	Instructions
Data transfer (7)	MOV, LDM, STM, XCH, SWAP, MOVTPE, MOVFPE
Arithmetic operations (17)	ADD, SUB, ADDS, SUBS, ADDX, SUBX, DADD, DSUB, MULXU, DIVXU, CMP, EXTS, EXTU, TST, NEG, CLR, TAS
Logic operations (4)	AND, OR, XOR, NOT
Shift (8)	SHAL, SHAR, SHLL, SHLR, ROTL, ROTR, ROTXL, ROTXR
Bit manipulation (4)	BSET, BCLR, BTST, BNOT
Branching (11)	Bcc*, JMP, PJMP, BSR, JSR, PJSR, RTS, PRTD, PRTS, RTD, SCB (/F, /NE, /EQ)
System control (12)	TRAPA, TRAP/VS, RTE, SLEEP, LDC, STC, ANDC, ORC, XORC, NOP, LINK, UNLK

Note: * Bcc is a conditional branch instruction in which cc represents a condition code.

This section is a concise summary of the instructions in each functional category. The MOV, ADD, and CMP instructions (table 3.10) have special short formats. For detailed descriptions of the instructions, refer to the *H8/500 Series Programming Manual*.

Operation Notation

Rd	General register (destination)
Rs	General register (source)
Rn	General register
(EAd)	Destination operand
(EAs)	Source operand
CCR	Condition code register
N	N (negative) bit of CCR
Z	Z (zero) bit of CCR
V	V (overflow) bit of CCR
C	C (carry) bit of CCR
CR	Control register
PC	Program counter
CP	Code page register
SP	Stack pointer
FP	Frame pointer
#Imm	Immediate data
disp	Displacement
+	Addition
–	Subtraction
×	Multiplication
÷	Division
∧	Logical AND
∨	Logical OR
⊕	Logical exclusive-OR
→	Move
↔	Exchange
¬	Not

3.5.2 Data Transfer Instructions

Instruction		Size*	Function
Data transfer	MOV:G	B/W	(EAs) → (EAd), #Imm → (EAd)
	MOV:E	B	MOV moves data between two general registers, or between a general register and memory, or moves immediate data to a general register or memory.
	MOV:I	W	
	MOV:F	B/W	
	MOV:L	B/W	
	MOV:S	B/W	
	LDM	W	Stack → Rn (register list)
			LDM pops data from the stack to one or more registers.
	STM	W	Rn (register list) → stack
			STM pushes data onto the stack from one or more registers.
	XCH	W	Rs ↔ Rd
			XCH exchanges data between two general registers.
	SWAP	B	Rd (upper byte) ↔ Rd (lower byte)
			SWAP exchanges the upper and lower bytes in a general register.
	MOVTPE	B	Rn → (EAd)
			MOVTPE transfers data from a general register to memory in synchronization with the E clock.
	MOVFPE	B	(EAs) → Rd
			MOVFPE transfers data from memory to a general register in synchronization with the E clock.

Note: * B = byte; W = word.

3.5.3 Arithmetic operations

Instruction		Size*	Function
Arithmetic operations	ADD:G	B/W	Rd ± (EAs) → Rd, (EAd) ± #Imm → (EAd)
	ADD:Q	B/W	These instructions perform addition or subtraction on data in a general register and data in another general register or memory, or on immediate data and data in a general register or memory.
	SUB	B/W	
	ADDS	B/W	
	SUBS	B/W	
	ADDX	B/W	Rd ± (EAs) ± C → Rd
	SUBX	B/W	These instructions perform addition or subtraction with carry or borrow on data in a general register and data in another general register or memory, or on immediate data and data in a general register or memory.
	DADD	B	$(Rd)_{10} \pm (Rs)_{10} \pm C \rightarrow (Rd)_{10}$
	DSUB	B	These instructions perform decimal addition or subtraction on data in two general registers.
	MULXU	B/W	Rd × (EAs) → Rd
			MULXU performs 8-bit × 8-bit or 16-bit × 16-bit unsigned multiplication on data in a general register and data in another general register or memory, or on data in a general register and immediate data.
	DIVXU	B/W	Rd ÷ (EAs) → Rd
			DIVXU performs 16-bit ÷ 8-bit or 32-bit ÷ 16-bit unsigned division on data in a general register and data in another general register or memory, or on data in a general register and immediate data.
	CMP:G	B/W	Rn – (EAs), (EAd) – #Imm
	CMP:E	B	CMP compares data in a general register with data in another general register or memory, or with immediate data, or compares immediate data with data in memory.
	CMP:I	W	
	EXTS	B	(<bit 7> of <Rd>) → (<bits 15 to 8> of <Rd>)
			EXTS converts byte data in a general register to word data by extending the sign bit.
	EXTU	B	0 → (<bits 15 to 8> of <Rd>)
			EXTU converts byte data in a general register to word data by padding the register with zero bits.
	TST	B/W	(EAd) – 0
			TST compares a general register or memory contents with 0.

Instruction		Size*	Function
Arithmetic operations (cont)	NEG	B/W	0 – (EAd) → (EAd) NEG obtains the two's complement of a general register or memory contents.
	CLR	B/W	0 → (EAd) CLR clears a general register or memory contents to 0.
	TAS	B	(EAd) — 0, $(1)_2$ → (<bit 7> of <EAd>) TAS tests a general register or memory contents, then sets the most significant bit (bit 7) to 1.

Note: * B = byte; W = word.

3.5.4 Logic Operation Instructions

Instruction		Size*	Function
Logical operations	AND	B/W	Rd ∧ (EAs) → Rd AND performs a logical AND operation on a general register with another general register, memory, or immediate data.
	OR	B/W	Rd ∨ (EAs) → Rd OR performs a logical OR operation on a general register with another general register, memory, or immediate data.
	XOR	B/W	Rd ⊕ (EAs) → Rd XOR performs a logical exclusive-OR operation on a general register with another general register, memory, or immediate data.
	NOT	B/W	¬ (EAd) → (EAd) NOT obtains the one's complement of a general register or memory contents.

Note: * B = byte; W = word.

3.5.5 Shift Operation Instructions

Instruction		Size*	Function
Shift operations	SHAL	B/W	(EAd) shift → (EAd)
	SHAR	B/W	These instructions perform an arithmetic shift operation on a general register or memory contents.
	SHLL	B/W	(EAd) shift → (EAd)
	SHLR	B/W	These instructions perform a logical shift operation on a general register or memory contents.
	ROTL	B/W	(EAd) shift → (EAd)
	ROTR	B/W	These instructions rotate a general register or memory contents.
	ROTXL	B/W	(EAd) rotate through carry → (EAd)
	ROTXR	B/W	These instructions rotate a general register or memory contents through the C (carry) bit.

Note: * B = byte; W = word.

3.5.6 Bit Manipulation Instructions

Instruction		Size*	Function
Bit manipulations	BSET	B/W	¬ (<bit-no.> of <EAd>) → Z, 1 → (<bit-no.> of <EAd>) BSET tests a specified bit in a general register or memory, then sets the bit to 1. The bit is specified by a bit number given in immediate data or a general register.
	BCLR	B/W	¬ (<bit-no.> of <EAd>) → Z, 0 → (<bit-no.> of <EAd>) BCLR tests a specified bit in a general register or memory, then clears the bit to 0. The bit is specified by a bit number given in immediate data or a general register.
	BNOT	B/W	¬ (<bit-no.> of <EAd>) → Z, → (<bit-no.> of <EAd>) BNOT tests a specified bit in a general register or memory, then inverts the bit. The bit is specified by a bit number given in immediate data or a general register.
	BTST	B/W	¬ (<bit-no.> of <EAd>) → Z BTST tests a specified bit in a general register or memory. The bit is specified by a bit number given in immediate data or a general register.

Note: * B = byte; W = word.

3.5.7 Branching Instructions

Instruction		Size	Function
Branching	Bcc	—	This instruction causes a branch if the specified mnemonic is true.

Mnemonic	Description	Condition
BRA (BT)	Always (true)	True
BRN (BF)	Never (false)	False
BHI	High	$C \vee Z = 0$
BLS	Low or same	$C \vee Z = 1$
BCC (BHS)	Carry clear (High or same)	$C = 0$
BCS (BLO)	Carry set (low)	$C = 1$
BNE	Not equal	$Z = 0$
BEQ	Equal	$Z = 1$
BVC	Overflow clear	$V = 0$
BVS	Overflow set	$V = 1$
BPL	Plus	$N = 0$
BMI	Minus	$N = 1$
BGE	Greater or equal	$N \oplus V = 0$
BLT	Less than	$N \oplus V = 1$
BGT	Greater than	$Z \vee (N \oplus V) = 0$
BLE	Less or equal	$Z \vee (N \oplus V) = 1$

	JMP	—	This instruction causes an unconditional branch to a specified address in the same page.
	PJMP	—	This instruction causes an unconditional branch to a specified address in a specified page.
	BSR	—	This instruction causes a branch to a subroutine at a specified address in the same page.
	JSR	—	This instruction causes a branch to a subroutine at a specified address in the same page.
	PJSR	—	This instruction causes a branch to a subroutine at a specified address in a specified page.
	RTS	—	This instruction causes a return from a subroutine in the same page.
	PRTS	—	This instruction causes a return from a subroutine in a different page.
	RTD	—	This instruction causes a return from a subroutine in the same page and adjusts the stack pointer.

Instruction		Size	Function
Branching (cont)	PRTD	—	This instruction causes a return from a subroutine in a different page and adjusts the stack pointer.
	SCB/F	—	These instructions control a loop using a loop counter and/or a specified termination condition.
	SCB/NE	—	
	SCB/EQ	—	

3.5.8 System Control Instructions

Instruction		Size	Function
System control	TRAPA	—	TRAPA generates a trap exception with a specified vector number.
	TRAP/VS	—	TRAP/VS generates a trap exception if the V bit is set to 1 when the instruction is executed.
	RTE	—	RTE causes a return from an exception-handling routine.
	LINK	—	FP → @–SP; SP → FP; SP + #Imm → SP LINK creates a stack frame.
	UNLK	—	FP → SP; @SP+ → FP UNLK deallocates a stack frame created by the LINK instruction.
	SLEEP	—	SLEEP causes a transition to the power-down state.
	LDC	B/W*	(EAs) → CR LDC moves immediate data, a general register, or memory contents to a specified control register.
	STC	B/W*	CR → (EAd) STC moves control register data to a specified general register or memory location.
	ANDC	B/W*	CR ∧ #Imm → CR ANDC logically ANDs a control register with immediate data.
	ORC	B/W*	CR ∨ #Imm → CR ORC logically ORs a control register with immediate data.
	XORC	B/W*	CR ⊕ #Imm → CR XORC logically exclusive-ORs a control register with immediate data.
	NOP	—	PC + 1 → PC No operation. This instruction only increments the program counter.

Note: * The size depends on the control register.

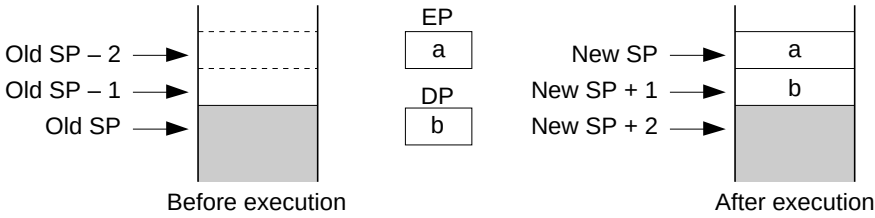
Note the following point when using the LDC and STC instructions to stack and unstack the BR, CCR, TP, DP, and EP control registers.

H8/500 CPU components do not permit byte-size access to the stack. If the LDC.B or STC.B assembler mnemonic is coded with the @R7+ (@SP+) or @–R7 (@–SP) addressing mode, the stack-pointer addressing mode takes precedence and hardware automatically performs word access. Specifically, the LDC.B and STC.B instructions are executed as shown in the following examples.

These examples only apply to the stack-pointer addressing modes. In addressing modes that do not use the stack pointer, byte-size data access is performed as specified by the assembler mnemonic.

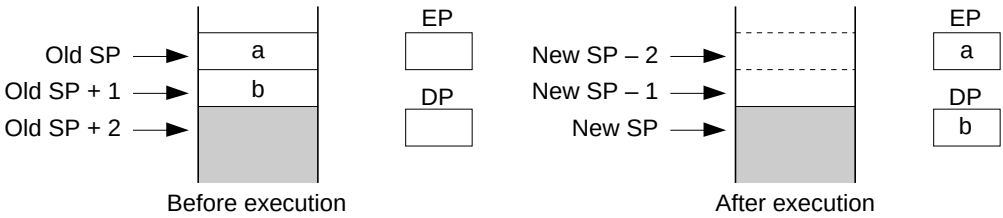
- `STC. B EP,@-SP`

  When word-size data access is applied to EP, both EP and DP are accessed. This instruction stores EP at address old SP – 2, and DP at address old SP – 1.



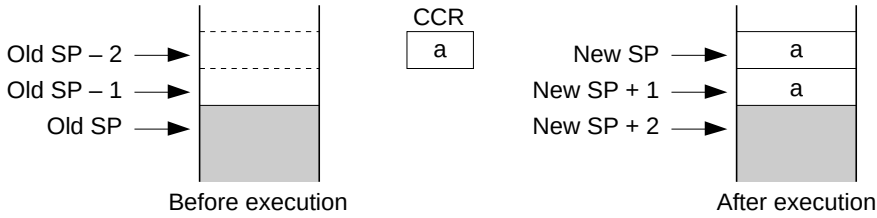
- `LDC.B  @SP+,EP`

  When word-size data access is applied to EP, both EP and DP are accessed. This instruction loads EP from address old SP, and DP from address old SP + 1, updating the DP value as well as the EP value.



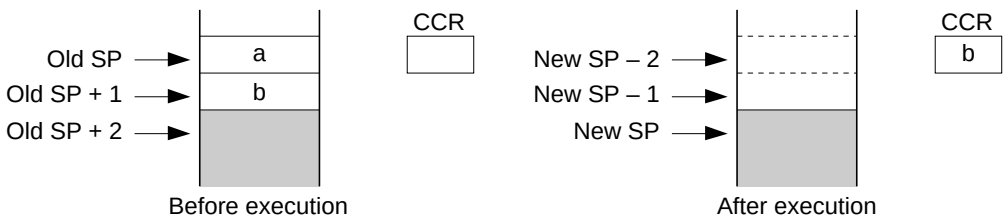
- `STC.B CCR,@-SP`

  When word-size data access is applied to CCR, only CCR is accessed. This instruction stores identical CCR contents at both address old SP – 2 and address old SP – 1.



- `LDC.B @SP+,CCR`

  When word-size data access is applied to CCR, only CCR is accessed. This instruction loads CCR from address old SP + 1. Take note that the value in address old SP will not be loaded.



  BR, DP, and TP are accessed in the same way as CCR. When DP is specified, both EP and DP are accessed, but when CCR, BR, DP, or TP is specified, only the specified register is accessed.

3.5.9 Short-Format Instructions

The ADD, CMP, and MOV instructions have special short formats (table 3.10).

The short formats are a byte shorter than the corresponding general formats, and most of them execute one state faster.

Table 3.10 Short-Format Instructions and Equivalent General Formats

Short-Format Instruction	Length	Equivalent Execution States*[2]	General-Format Instruction	Length	Execution States*[2]
ADD:Q #xx,Rd*[1]	2	2	ADD:G #xx:8,Rd	3	3
CMP:E #xx:8,Rd	2	2	CMP:G.B #xx:8,Rd	3	3
CMP:I #xx:16,Rd	3	3	CMP:G.W #xx:16,Rd	4	4
MOV:E #xx:8,Rd	2	2	MOV:G.B #xx:8,Rd	3	3
MOV:I #xx:16,Rd	3	3	MOV:G.W #xx:16,Rd	4	4
MOV:L @aa:8,Rd	2	5	MOV:G @aa:8,Rd	3	5
MOV:S Rs,@aa:8	2	5	MOV:G Rs,@aa:8	3	5
MOV:F @(d:8,R6),Rd	2	5	MOV:G @(d:8,R6),Rd	3	5
MOV:F Rs,@(d:8,R6)	2	5	MOV:G Rs,@(d:8,R6)	3	5

Notes: 1. The ADD:Q instruction accepts other destination operands in addition to a general register, but the immediate data value (#xx) is limited to ±1 or ±2.
2. Number of execution states for access to memory.

3.6 Operating Modes

The CPU operation has two modes characterized by the memory size: the minimum mode and the maximum mode. It also has two other modes characterized by the data bus size: the 8-bit data bus mode and the 16-bit data bus mode.

The combination of these modes is selected by the mode pins (MD_2 to MD_0).

3.6.1 Minimum Mode

The minimum mode supports a maximum address space of 64 kbytes. The page registers are ignored, and instructions that branch across page boundaries (PJMP, PJSR, PRTS, PRTD) are invalid.

3.6.2 Maximum Mode

In the maximum mode the page registers can be used with a maximum address space of 1 Mbyte. $P5_3$ to $P5_0$ of port 5 are used for address bus A_{19} to A_{16}.

The address space is divided into separate 64-kbyte pages. Since the pages are separate, it is not possible to move continuously across a page boundary.

However, it is possible to move from one page to another using branching instructions (PJMP, PJSR, PRTS, PRTD). The TRAPA instruction can branch to interrupt-handling routines and also jump across page boundaries. It is not necessary for a program to be contained on a single 64-kbyte page.

When data-size access crosses a page boundary, the page register must be rewritten by the program before it can access the data in the next page.

3.6.3 8-Bit Data Bus Mode

The data bus in this mode is 8 bits wide. The basic cycle of an external access consists of 3 cycles.

3.6.4 16-Bit Data Bus Mode

The data bus in this mode is 16 bits wide. Port 12 is used for data bus D_{15} to D_8. The basic cycle of an external access consists of 2 or 3 cycles.

3.7 Basic Operational Timing

3.7.1 Overview

The CPU operates on a system clock (ø) which is created by dividing an oscillator frequency (f_{OSC}) by two. One period of the system clock is referred to as a “state.” The CPU accesses memory in a cycle consisting of 2 or 3 of these states. The CPU uses different methods to access RAM, the register field, and external devices.

3.7.2 Access to Memory (RAM)

For maximum speed, access to memory is performed in two states, using a 16-bit-wide data bus. The bus control signals output from the H8/570 MCU switch to the nonactive state during accessing.

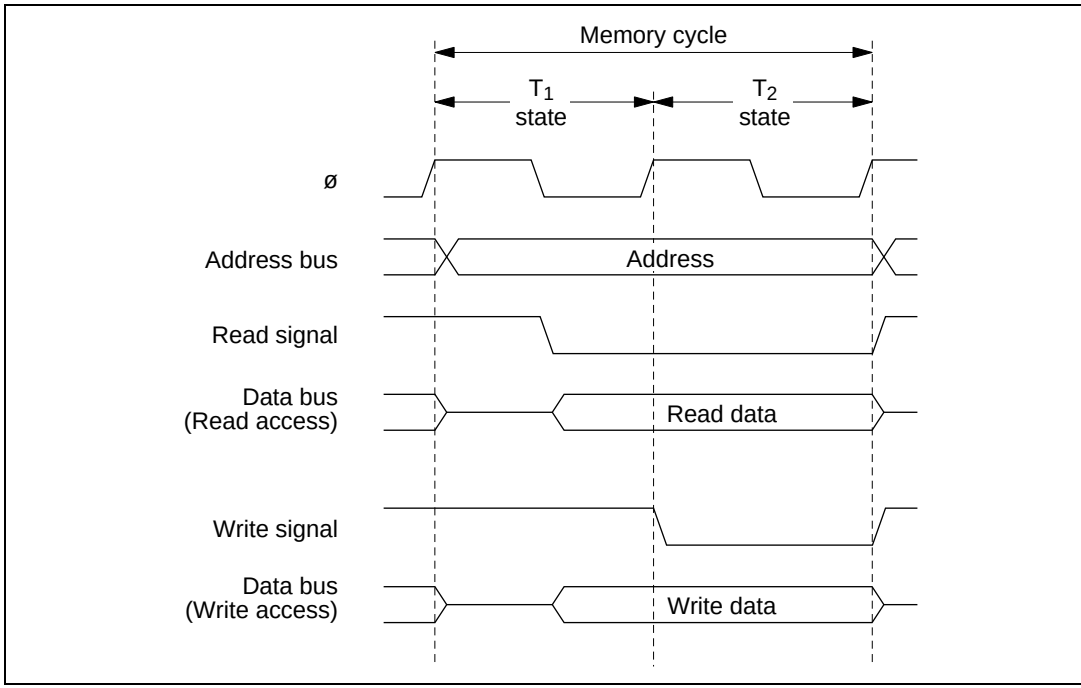


Figure 3.6 Memory Access Timing

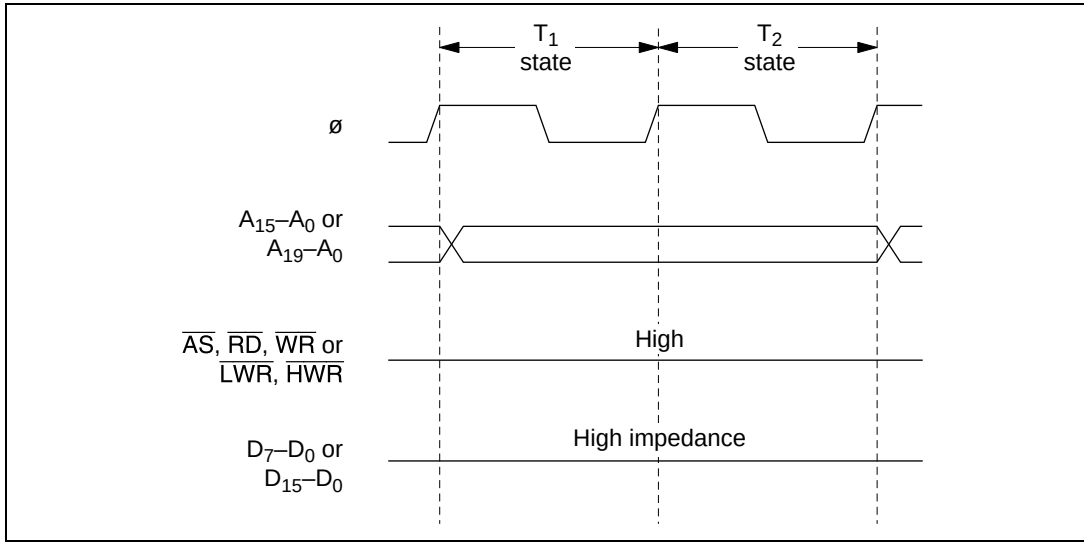


Figure 3.7 Pin States During Access to Memory

3.7.3 Access to Register Field

The access cycle for accessing the register field consists of three states. The data bus for this access can be either 8 or 16 bits wide.

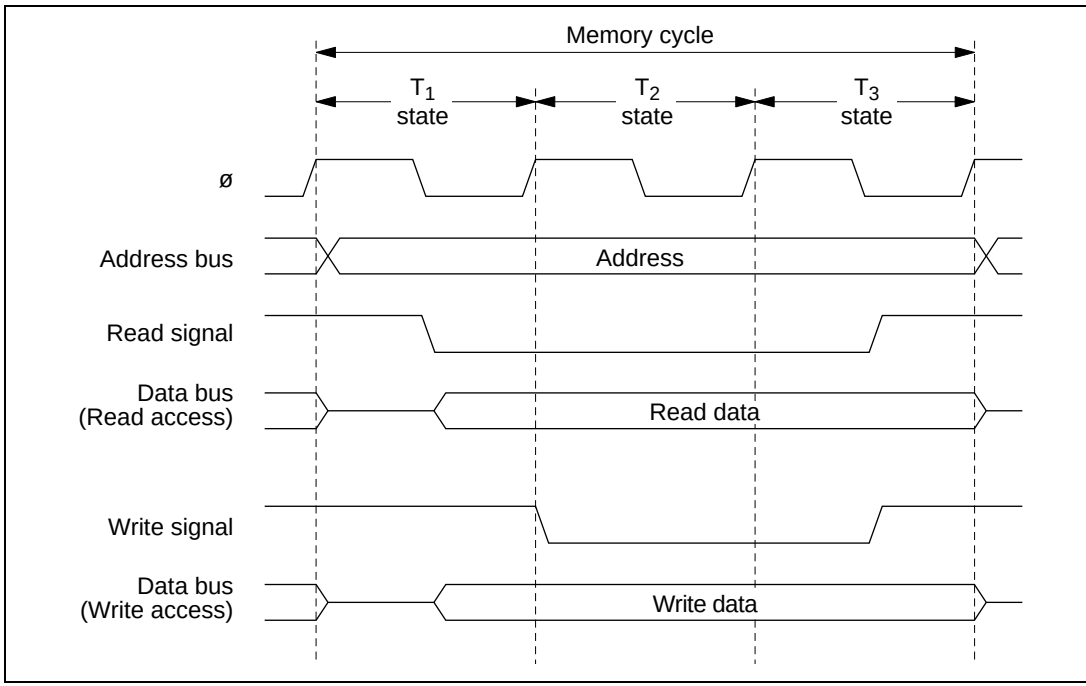


Figure 3.8 Register Field Access Timing

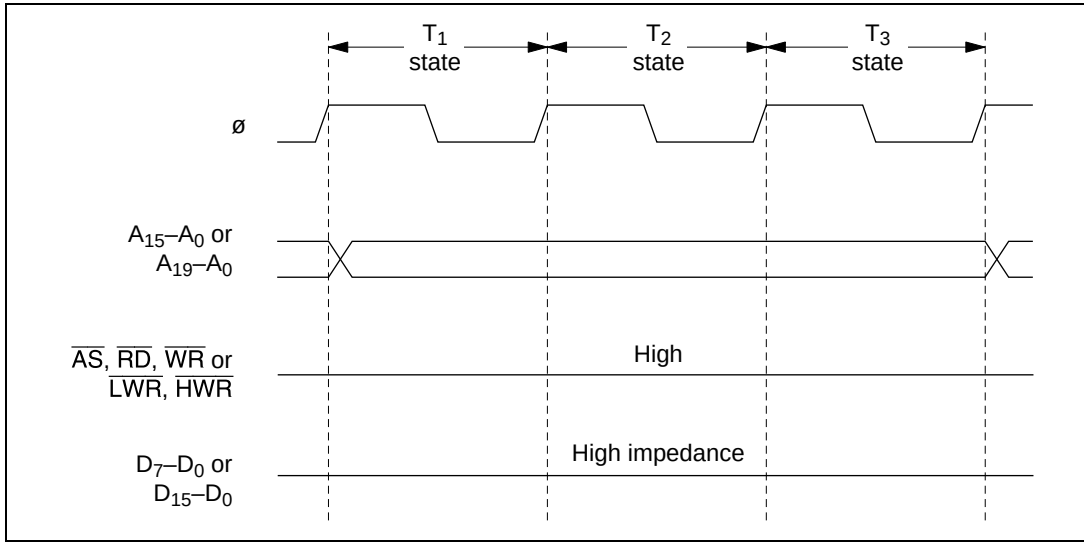


Figure 3.9 Pin States During Register Field Access

3.7.4 Access to External Three-State Access Space

This access is used for interfacing with low-speed devices. Additional wait states (T_w) can be inserted by the wait-state controller (WSC).

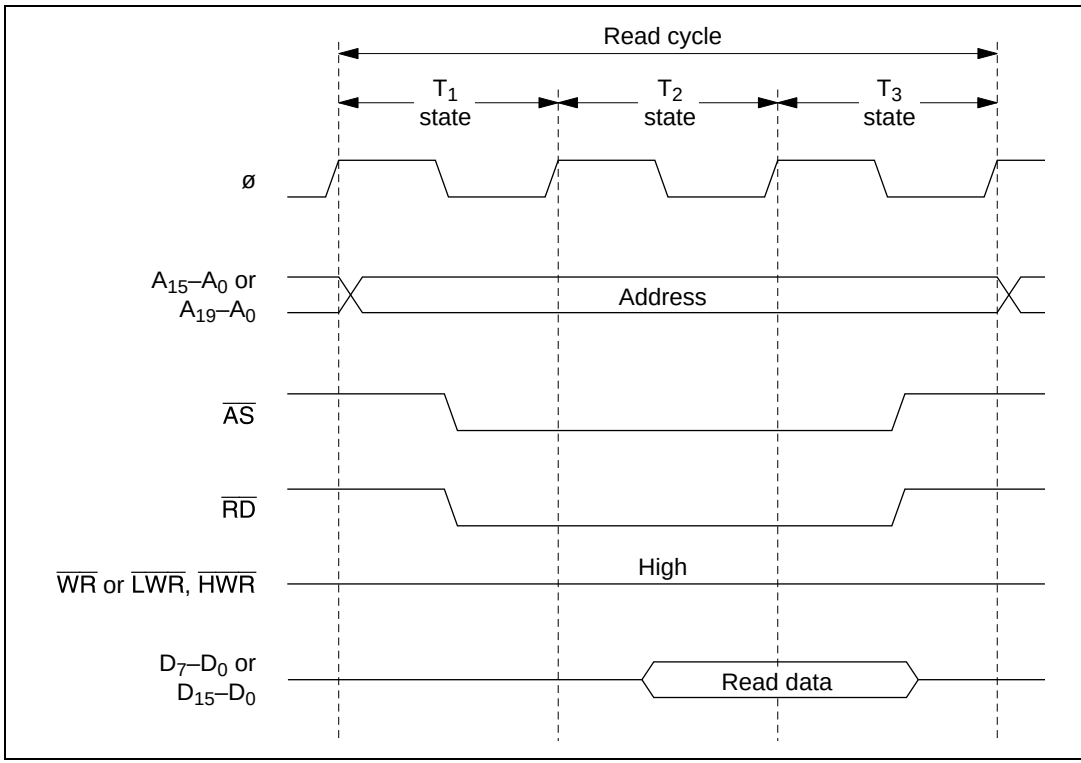


Figure 3.10 (a) Access Cycle of Three-State Access Space (Read Access)

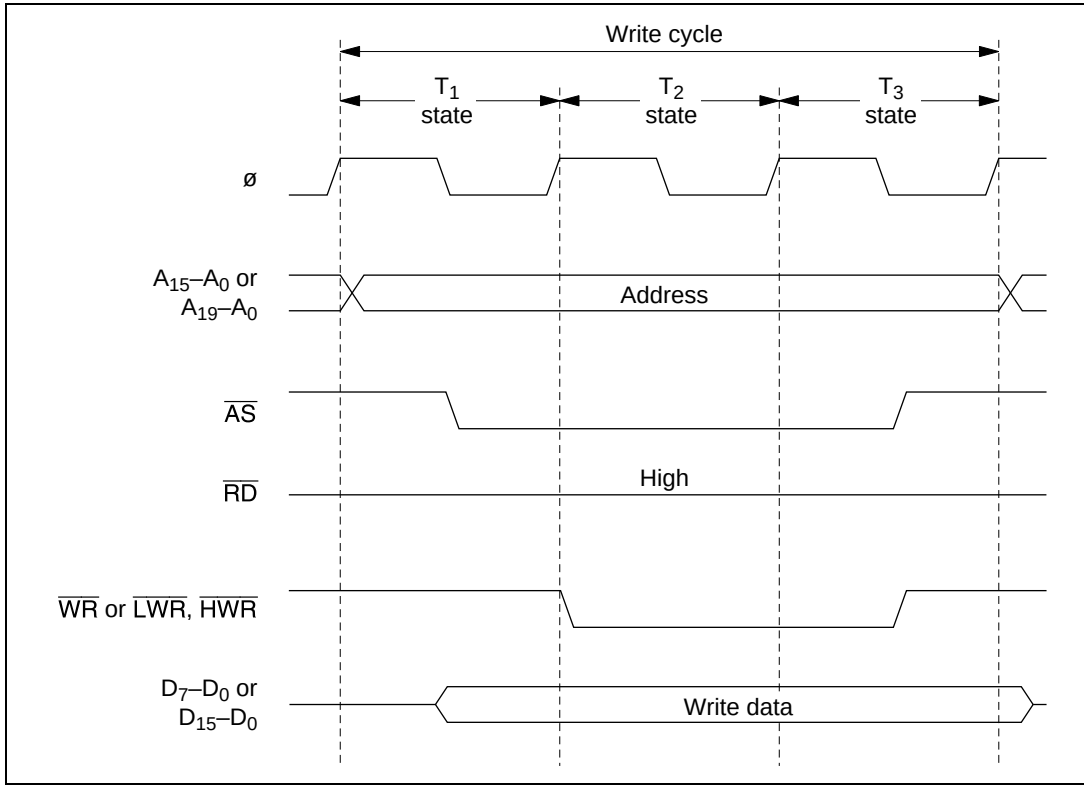


Figure 3.10 (b) Access Cycle of Three-State Access Space (Write Access)

3.7.5 Access to External Two-State Word-Wide Space

This access is used for maximum speed. Additional wait states (T_w) cannot be inserted.

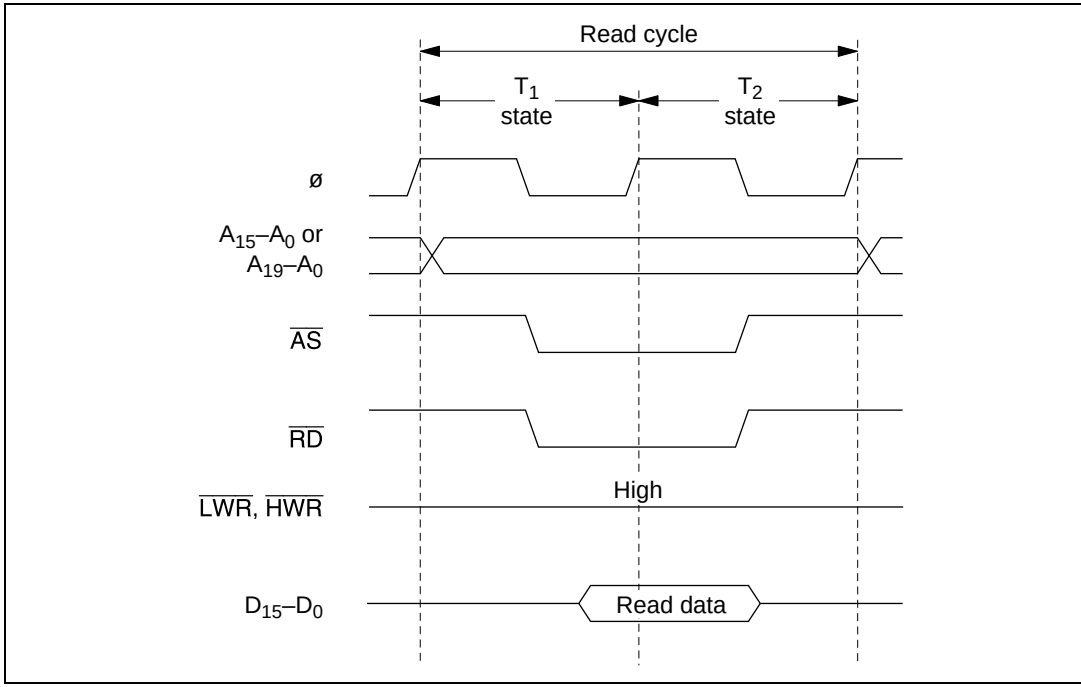


Figure 3.11 (a) Access Cycle of Two-State Access Space (Read Access)

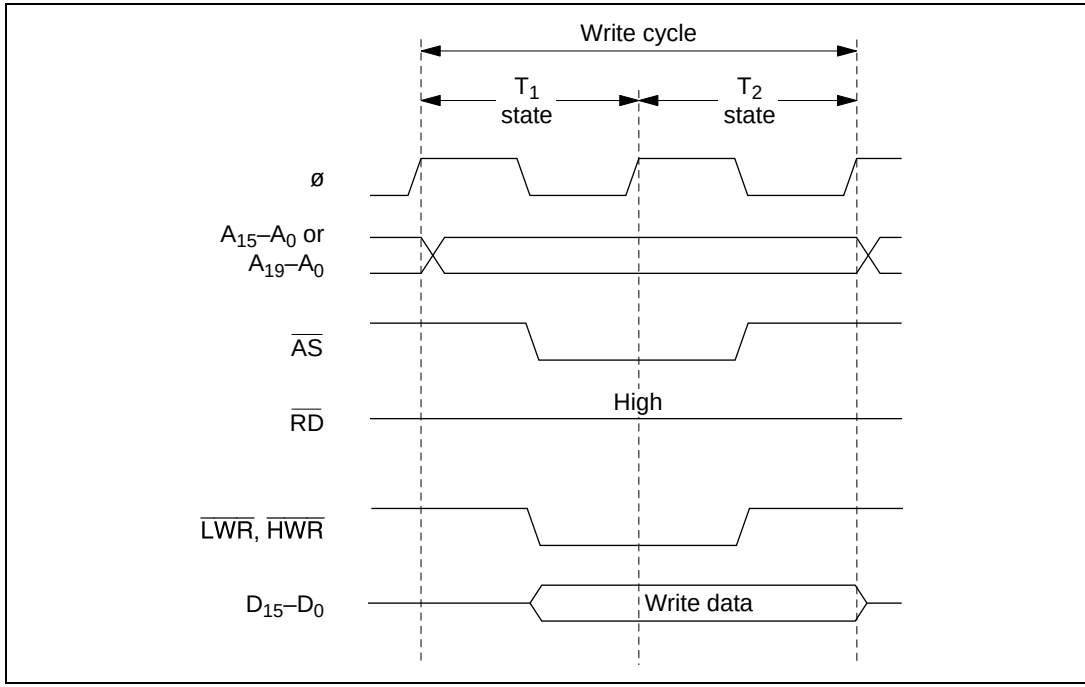


Figure 3.11 (b) Access Cycle of Two-State Access Space (Write Access)

3.8 CPU States

3.8.1 Overview

The CPU has five states: program execution state, exception-handling state, bus-released state, reset state, and power-down state. The power-down state is further divided into the sleep mode, software standby mode, and hardware standby mode. See figure 3.12.

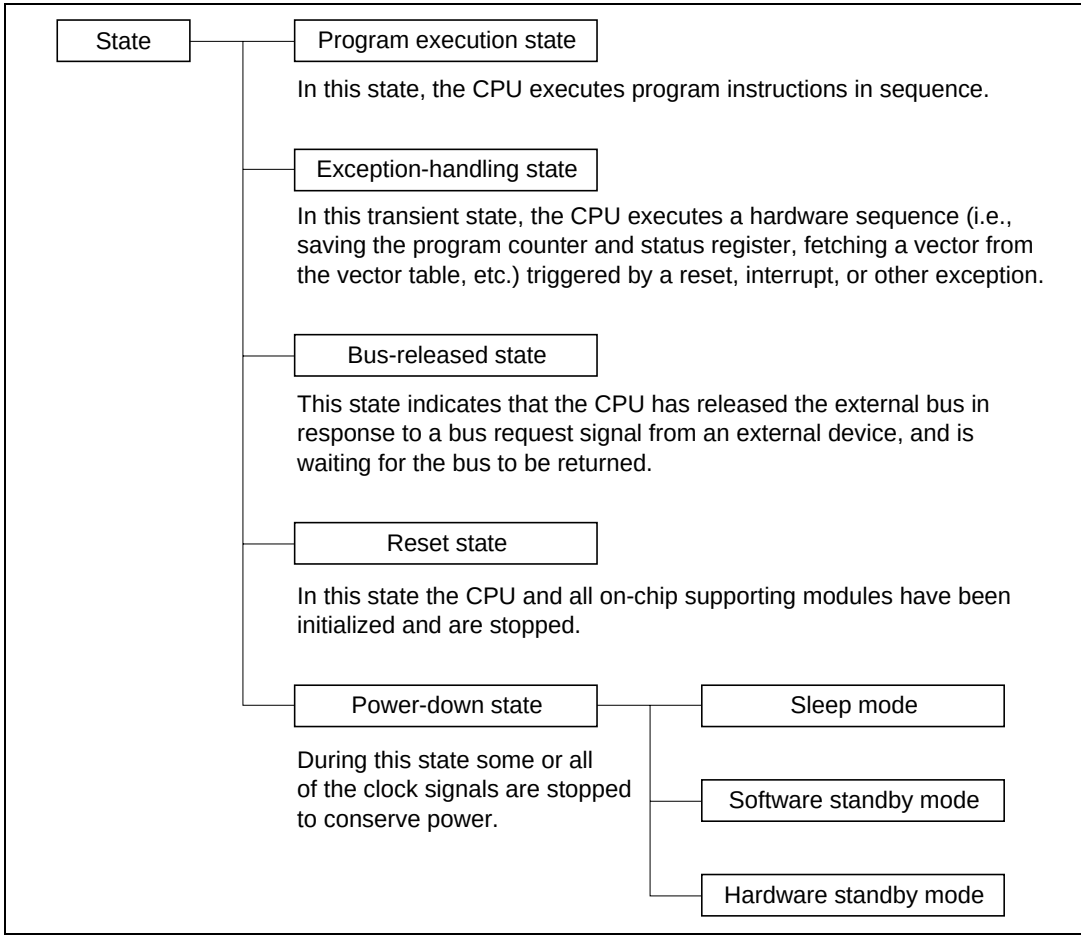
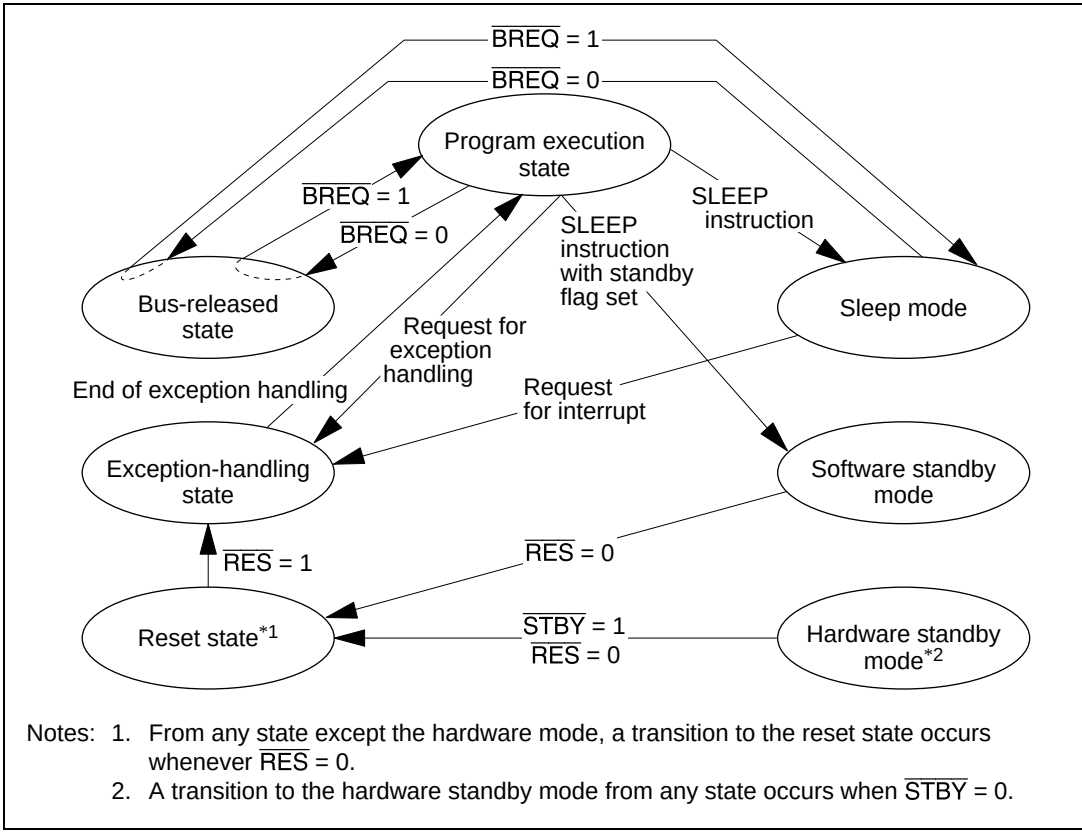

State
Program execution state
In this state, the CPU executes program instructions in sequence.
Exception-handling state
In this transient state, the CPU executes a hardware sequence (i.e., saving the program counter and status register, fetching a vector from the vector table, etc.) triggered by a reset, interrupt, or other exception.
Bus-released state
This state indicates that the CPU has released the external bus in response to a bus request signal from an external device, and is waiting for the bus to be returned.
Reset state
In this state the CPU and all on-chip supporting modules have been initialized and are stopped.
Power-down state
During this state some or all of the clock signals are stopped to conserve power.
Sleep mode
Software standby mode
Hardware standby mode

Figure 3.12 Operating States



Notes: 1. From any state except the hardware mode, a transition to the reset state occurs whenever $\overline{RES}$ = 0.
2. A transition to the hardware standby mode from any state occurs when $\overline{STBY}$ = 0.

Figure 3.13 State Transitions

3.8.2 Program Execution State

In this state the CPU executes program instructions in normal sequence.

3.8.3 Exception-Handling State

The exception-handling state is a transient state that occurs when the CPU alters the normal program flow due to an interrupt, trap instruction, address error, or other exception. In this state the CPU carries out a hardware-controlled sequence that enables it to execute a user-coded exception-handling routine.

In the hardware exception-handling sequence the CPU does the following:

- Saves the program counter and status register (in minimum mode), or program counter, code page register, and status register (in maximum mode) to the stack
- Clears the T bit in the status register to 0

- Fetches the start address of the exception-handling routine from the exception vector table
- Branches to the start address, thus returning to the program execution state

See section 4, Exception Handling, for further information on the exception-handling state.

3.8.4 Bus-Released State

When the external bus is requested, the CPU can grant ownership of the bus to an external device. While an external device has the bus-access right, the CPU is said to be in the bus-released state. The bus-access right is controlled by two pins:

$\overline{\text{BREQ}}$: Input pin for the bus request signal from an external device
$\overline{\text{BACK}}$: Output pin for the bus request acknowledge signal from the CPU, indicating that the CPU has released the bus

The CPU enters and leaves the bus-released state by the following procedure:

1. The CPU receives an asserted low $\overline{\text{BREQ}}$ signal from an external device.
2. The CPU then places the address bus pins (A_{19}–A_0), data bus pins (D_7–D_0) and bus control pins ($\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{HWR}}$, $\overline{\text{LWR}}$, and $\overline{\text{AS}}$) in the high-impedance state, drives the $\overline{\text{BACK}}$ pin low to indicate that it has released the bus, and halts.
3. The external device that requested the bus (by $\overline{\text{BREQ}}$) becomes the bus master, enabling it to use the data bus and address bus. The external device is thus in control of manipulating the bus control signals ($\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{HWR}}$, $\overline{\text{LWR}}$, and $\overline{\text{AS}}$).
4. To release the bus from the external device, $\overline{\text{BREQ}}$ must be driven high by the external device itself. The CPU then reassumes control of the bus and returns to the program execution state.

Bus Release Timing: The CPU can release the bus-access right at the following times:

- Since $\overline{\text{BREQ}}$ is sampled during every memory access cycle (instruction prefetch or data read/write), the CPU is able to release the bus-access right at the end of the cycle whenever $\overline{\text{BREQ}}$ is low. (In the case of word-size data access to external memory or to an address from H'FE80 to H'FFFF, the CPU does not release the bus-access right until it has accessed both the upper and lower data bytes.)
- $\overline{\text{BREQ}}$ is also sampled at internal machine cycles during the execution of the MULXU and DIVXU instructions, since considerable time may pass without an instruction prefetch or data read/write. The bus-access right is released if $\overline{\text{BREQ}}$ is asserted low.
- The bus-access right can also be released in the sleep mode.

The CPU does not recognize interrupts while the bus is released.

Timing Charts: The timing charts of the bus release operations are shown in figures 3.14 (bus release during access cycle of memory or two-state access space), 3.15 (bus release during three-state access cycle), and 3.16 (bus release while the CPU is performing an internal operation).

Notes: $\overline{\text{BREQ}}$ must be held low until $\overline{\text{BACK}}$ is driven low. If $\overline{\text{BREQ}}$ is driven high before $\overline{\text{BACK}}$ is low, the bus release operation may be executed incorrectly.

To leave the bus-released state, a nonasserted high $\overline{\text{BREQ}}$ must be sampled twice consecutively. If $\overline{\text{BREQ}}$ is re-asserted low before it is sampled again, the bus released cycle will not end.

The bus release operation is enabled only when the BRLE bit in system control register 1 (SYSCR1) is set to 1. When this bit is cleared to 0 (its initial value), the $\overline{\text{BREQ}}$ and $\overline{\text{BACK}}$ pins are used as general-purpose input and output, $P1_3$ and $P1_2$, respectively.

An example instruction that can set the BRLE bit is:

```
BSET.B #3,@H'FFFC
```

3.8.5 Reset State

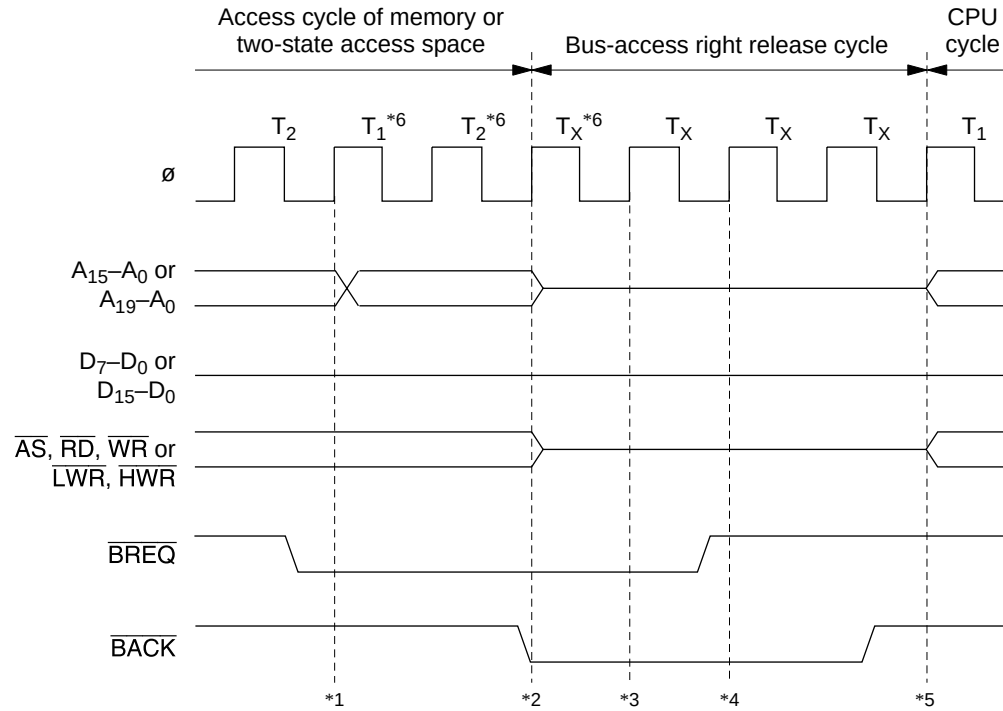
In the reset state, the CPU and all on-chip supporting modules are initialized and halted. The CPU enters the reset state whenever the $\overline{\text{RES}}$ pin is asserted low, unless the CPU is currently in the hardware standby mode. The CPU remains in the reset state until the $\overline{\text{RES}}$ pin is driven high.

See section 4.2, Reset, for further information on the reset state.

3.8.6 Power-Down State

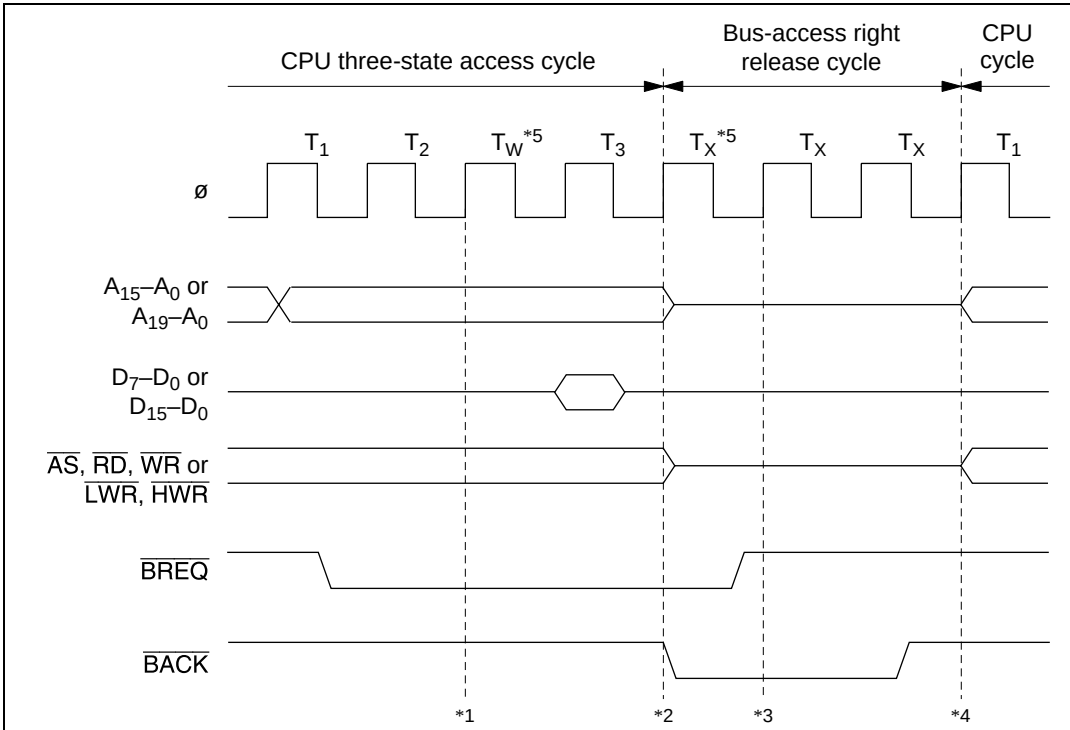
The power-down state consists of three modes: the sleep mode, the software standby mode, and the hardware standby mode.

See section 16, Power-Down State, for further information.



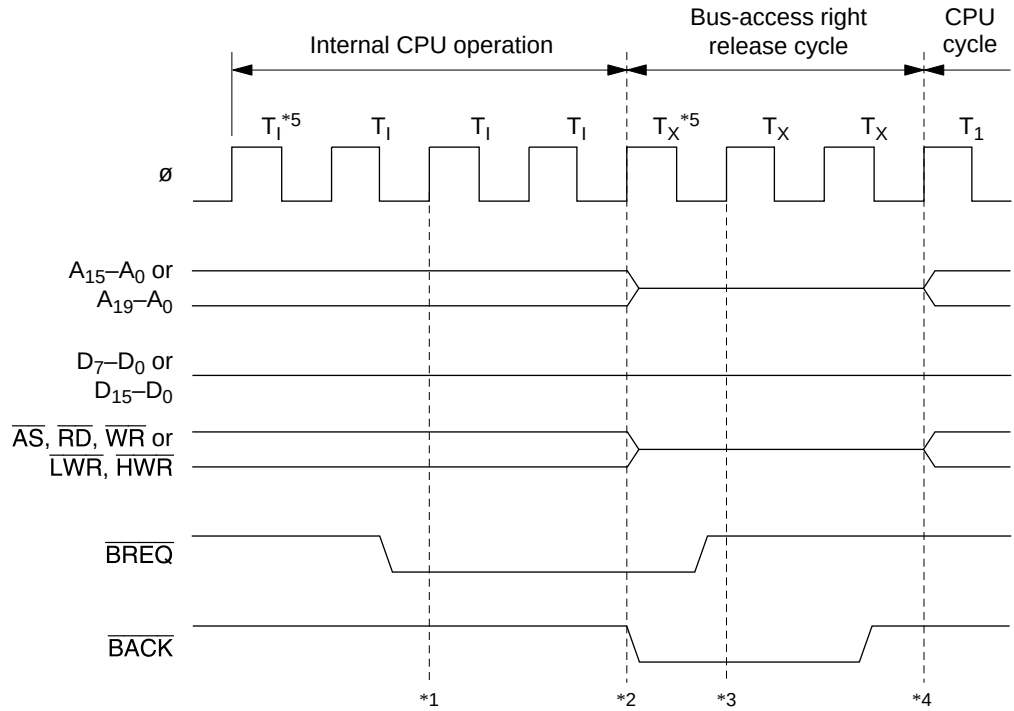
Notes: 1. The $\overline{\text{BREQ}}$ pin is sampled at the start of the T_1 state and an asserted low $\overline{\text{BREQ}}$ is detected.
2. At the end of the memory access cycle, the $\overline{\text{BACK}}$ pin is driven low and the CPU releases the bus.
3. While the bus is released, the $\overline{\text{BREQ}}$ pin is sampled at each T_X state.
4. A nonasserted high $\overline{\text{BREQ}}$ is detected.
5. The $\overline{\text{BACK}}$ pin is driven high, ending the bus-access right release cycle.
6. T_1 and T_2: Memory access states.
T_X: Bus-access right released state.

Figure 3.14 Bus-Access Right Release Cycle (During Access Cycle of Memory or Two-State Access Space)



Notes: 1. The $\overline{\text{BREQ}}$ pin is sampled at the start of the T_W state and an asserted low $\overline{\text{BREQ}}$ is detected.
2. At the end of the external access cycle, the $\overline{\text{BACK}}$ pin is driven low and the CPU releases the bus.
3. The $\overline{\text{BREQ}}$ pin is sampled at the T_X state and a nonasserted high $\overline{\text{BREQ}}$ is detected.
4. The $\overline{\text{BACK}}$ pin is driven high, ending the bus-access right release cycle.
5. T_W: Wait state.
T_X: Bus-access right released state.

Figure 3.15 Bus-Access Right Release Cycle (During Three-State Access Cycle)



Notes: 1. The $\overline{BREQ}$ pin is sampled at the start of a T_I state and an asserted low $\overline{BREQ}$ is detected.
2. At the end of the internal operation cycle, the $\overline{BACK}$ pin is driven low and the CPU releases the bus.
3. The $\overline{BREQ}$ pin is sampled at the T_X state and a nonasserted high $\overline{BREQ}$ is detected.
4. The $\overline{BACK}$ pin is driven high, ending the bus-access right release cycle.
5. T_I: Internal CPU operation state.
T_X: Bus-access right released state.

Figure 3.16 Bus-Access Right Release Cycle (During Internal CPU Operation)

Section 4 Exception Handling

4.1 Overview

4.1.1 Types of Exception Handling and Their Priority

Exception handling can be initiated by a reset, address error, trace, interrupt, or instruction (tables 4.1 (a) and (b)). An instruction initiates exception handling if the instruction is an invalid instruction, a trap instruction, or a DIVXU instruction with zero divisor. Exception handling begins with a hardware exception-handling sequence which prepares for the execution of a user-coded software exception-handling routine.

The priority order among the different types of exceptions is listed in table 4.1 (a). If two or more exceptions occur simultaneously, they are handled in their order of priority. An instruction exception cannot occur simultaneously with other types of exceptions.

Table 4.1 (a) Exceptions and Their Priority

	Exception Type	Source	Detection Timing	Start of Exception-Handling Sequence
High	Reset	External	$\overline{\text{RES}}$ low-to-high transition	Immediately
↑	Address error	Internal	Instruction fetch or data read/write bus cycle	End of instruction execution
↑	Trace	Internal	End of instruction execution, if T = 1 in status register	End of instruction execution
↑	Interrupt	External, internal	End of instruction execution or end of exception-handling sequence	End of instruction execution
Low				

Table 4.1 (b) Instruction Exceptions

Exception Type	Start of Exception-Handling Sequence
Invalid instruction	Attempted execution of instruction with undefined code
Trap instruction	Started by execution of trap instruction
Zero divide	Attempted execution of DIVXU instruction with zero divisor

4.1.2 Hardware Exception-Handling Sequence

The hardware exception-handling sequence varies depending on the type of exception. When exception handling is initiated by a factor other than reset, the CPU will:

1. Save the program counter and status register (in minimum mode), or program counter, code page register, and status register (in maximum mode) to the stack.
2. Clear the T bit in the status register to 0.
3. Fetch the start address of the exception-handling routine from the exception vector table.
4. Branch to the start address.

For an interrupt, the CPU also alters the interrupt mask level in bits I_2 to I_0 of the status register.

For reset, step (1) is omitted. See section 4.2, Reset, for the full reset sequence.

4.1.3 Exception Factors and Vector Table

The factors that initiate exception handling can be classified (figure 4.1).

The starting addresses of the exception-handling routines for each factor are contained in an exception vector table located in the low addresses of page 0. See table 4.2. Note that there are different addresses for the minimum and maximum modes.

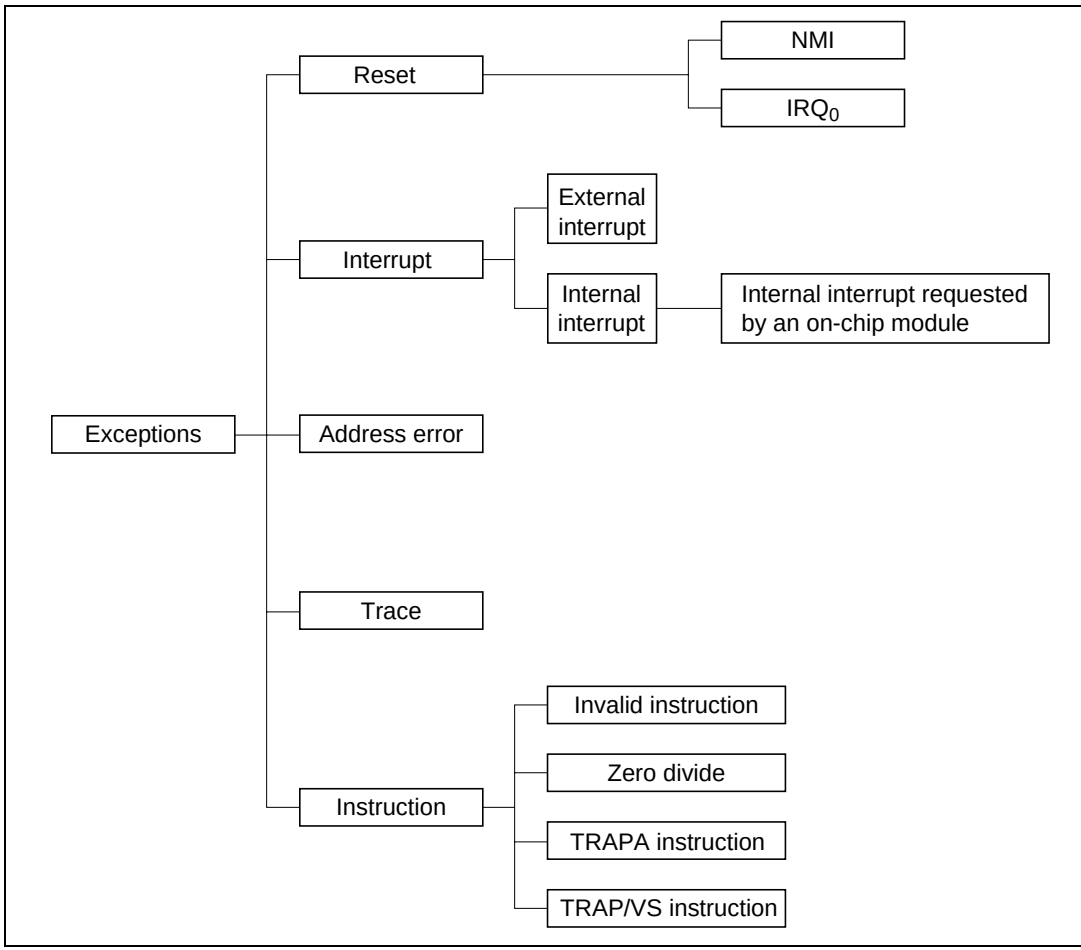


Figure 4.1 Types of Factors Causing Exception Handling

Table 4.2 Exception Vector Table

	Vector Address	
Type of Exception	**Minimum Mode**	**Maximum Mode*[1]**
Reset (Initialize PC)	H'0000 to H'0001	H'0000 to H'0003
— (Reserved for system)	H'0002 to H'0003	H'0004 to H'0007
Invalid instruction	H'0004 to H'0005	H'0008 to H'000B
DIVXU instruction (Zero divide)	H'0006 to H'0007	H'000C to H'000F
TRAP/VS instruction	H'0008 to H'0009	H'0010 to H'0013
— (Reserved for system)	H'000A–H'000B to H'000E–H'000F	H'0014–H'0017 to H'001C–H'001F
Address error	H'0010 to H'0011	H'0020 to H'0023
Trace	H'0012 to H'0013	H'0024 to H'0027
— (Reserved for system)	H'0014 to H'0015	H'0028 to H'002B
Nonmaskable external interrupt (NMI)	H'0016 to H'0017	H'002C to H'002F
— (Reserved for system)	H'0018–H'0019 to H'001C–H'001D	H'0030–H'0033 to H'0038–H'003B
ISP address error	H'001E to H'001F	H'003C to H'003F
TRAPA instruction (16 vectors)	H'0020–H'0021 to H'003E–H'003F	H'0040–H'0043 to H'007C–H'007F
External interrupts, IRQ_0	H'0040 to H'0041	H'0080 to H'0083
Internal interrupts*[2]	H'0044–H'0045 to H'0078–H'0079	H'0088–H'008B to H'00F0–H'00F3

Notes: 1. The exception vector table is located at the beginning of page 0.
2. For details of the internal interrupt vectors, refer to table 5.2.

4.2 Reset

4.2.1 Overview

Reset has the highest exception-handling priority.

When the $\overline{\text{RES}}$ pin is asserted low, all current processing is halted and the H8/570 MCU enters the reset state.

Reset initializes the internal status of the CPU and the registers of the on-chip supporting modules and I/O ports. It does not initialize the RAM.

When the $\overline{\text{RES}}$ pin is driven from low to high, the H8/570 MCU comes out of the reset state and begins executing the hardware reset sequence.

4.2.2 Reset Sequence

The reset signal is detected whenever the $\overline{\text{RES}}$ pin is asserted low.

However, to ensure that the H8/570 is reset, the $\overline{\text{RES}}$ pin should be held low for at least 20 ms at power-up. To reset the H8/570 during operation, the $\overline{\text{RES}}$ pin should be held low for at least 6ø clock cycles. See table E-1, Port State, in appendix E for the status of other pins in the reset state.

When the $\overline{\text{RES}}$ pin is driven high after being held low for the necessary time, the hardware reset exception-handling sequence begins, during which:

1. In the status register (SR), the T bit is cleared to disable the trace mode, and the interrupt mask level (bits I_2 to I_0) is set to 7. The reset disables all interrupts, including NMI.
2. The CPU loads the reset start address from the vector table into the program counter and begins executing the program at that address.

The contents of the vector table differs between minimum mode and maximum mode (figure 4.2). This affects step (2) as follows:

Minimum Mode: One word is copied from addresses H'0000 and H'0001 in the vector table to the program counter. Program execution then begins from the address in the program counter (PC). See figures 4.2 and 4.3.

Maximum Mode: Two words are read from addresses H'0000 to H'0003 in the vector table. The byte in address H'0000 is ignored. The byte in address H'0001 is copied to the code page register (CP). The contents of addresses H'0002 and H'0003 are copied to the program counter. Program execution starts from the address indicated by the code page register and program counter. See figures 4.2 and 4.4.

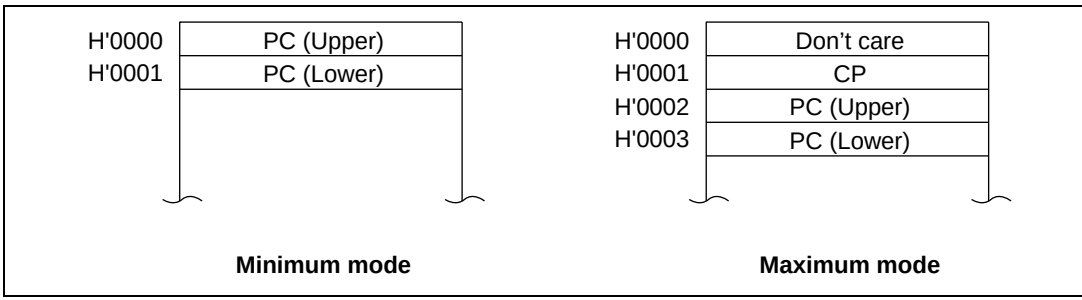
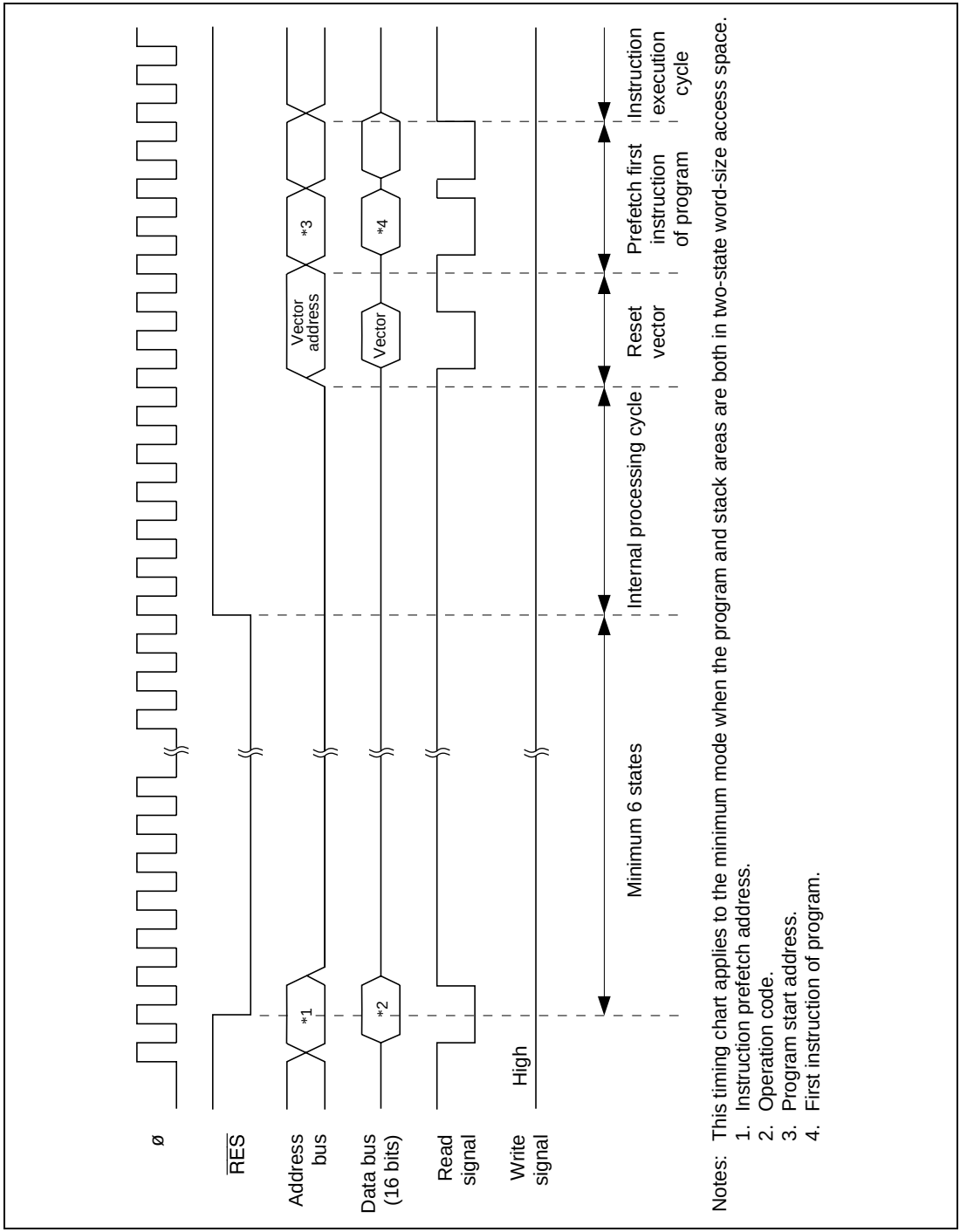


Figure 4.2 Reset Vector

4.2.3 Stack Pointer Initialization

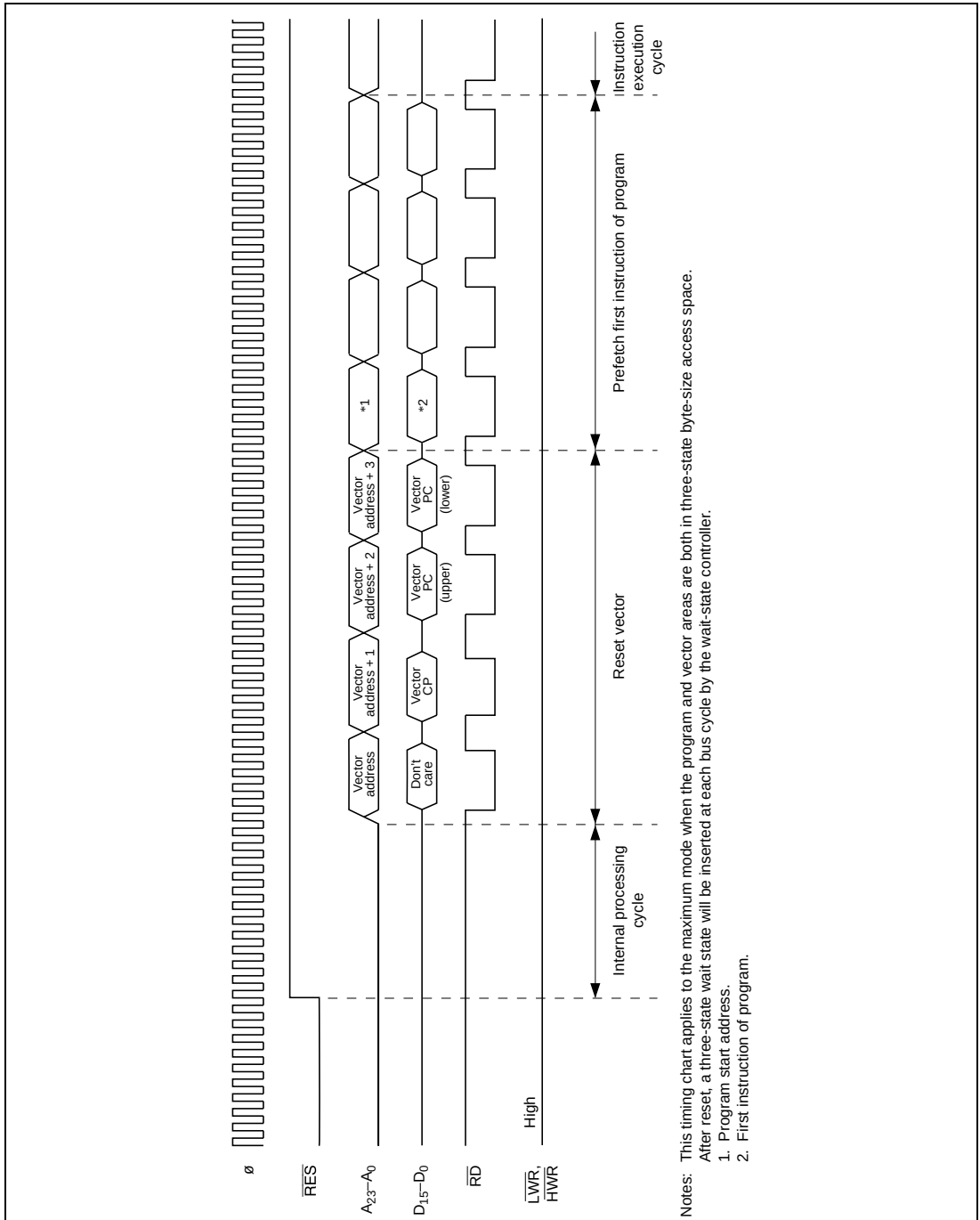
The hardware reset sequence does not initialize the stack pointer, so this must be done by software. If an interrupt was to be accepted after reset and before the stack pointer (SP) is initialized, the program counter and status register would not be saved correctly, causing a program crash. This can be avoided by coding the reset routine as explained in the following.

When the MCU comes out of the reset state all interrupts, including NMI, are disabled, so the instruction at the reset start address is always executed. In the minimum mode, this instruction should initialize the stack pointer (SP). In the maximum mode, this instruction should be an LDC instruction initializing the stack page register (TP), and the next instruction should initialize the stack pointer. Execution of the LDC instruction disables interrupts again, ensuring that the stack pointer initializing instruction is executed.



Notes: This timing chart applies to the minimum mode when the program and stack areas are both in two-state word-size access space.
1. Instruction prefetch address.
2. Operation code.
3. Program start address.
4. First instruction of program.

Figure 4.3 Reset Sequence (Minimum Mode)



Notes: This timing chart applies to the maximum mode when the program and vector areas are both in three-state byte-size access space.
After reset, a three-state wait state will be inserted at each bus cycle by the wait-state controller.
1. Program start address.
2. First instruction of program.

Figure 4.4 Reset Sequence (Maximum Mode)

4.3 Address Error

There are two causes of address errors:

- Illegal instruction prefetch
- Word-size data access at an odd address

An address error initiates the address error exception-handling sequence. This sequence clears the T bit of the status register to 0 to disable the trace mode, but does not affect the interrupt mask level in bits I_2 to I_0.

4.3.1 Illegal Instruction Prefetch

An attempt to prefetch an instruction from the register field or from an 8-bit I/O device space in memory addresses H'FE80 to H'FFFF and H'F0000 to H'FFFFF in modes 3 and 6 causes an address error regardless of the MCU operating mode.

Handling of this type of address error begins when the prefetch cycle that caused the error has been completed and the execution of the current instruction has also been completed. The program counter value pushed onto the stack is the address of the instruction immediately following the last instruction executed.

Program code should not be located in addresses H'FE7D to H'FE7F. If the CPU executes an instruction in these addresses, it will attempt to prefetch the next instruction from the register field, causing an address error.

4.3.2 Word-Size Data Access at an Odd Address

If an attempt is made to access word data starting at an odd address, an address error occurs regardless of the MCU operating mode. The program counter value pushed onto the stack during the handling of this error is the address of the next instruction after the instruction that attempted the illegal word access.

4.4 Trace

When the T bit of the status register is set to 1, the CPU operates in trace mode. In this mode, a trace exception occurs at the completion of each instruction. The trace mode can be used to debug a program for a debugger.

In the trace exception sequence the T bit of the status register is cleared to 0 to disable the trace mode while the trace routine is executing. The interrupt mask level in bits I_2 to I_0 does not change.

Within the status-register data saved on the stack, the T bit is set to 1. When the trace routine returns by the RTE instruction, the status register is popped from the stack and the trace mode resumes.

4.5 Interrupts

Interrupts can be requested from two external sources (NMI and IRQ_0) and five on-chip supporting modules: the PWM (pulse width modulation) timer, the serial communication interface (SCI), the A/D converter, the Intelligent Sub-Processor (ISP), and the watchdog timer (WDT). The internal interrupt sources can request a total of twenty-four different types of interrupts, each having its own interrupt vector. See figure 4.5.

Each interrupt source has a priority. An NMI interrupt has the highest priority, and is normally accepted unconditionally. The priorities of the other interrupt sources are set in control registers (IPRA to IPRD) in the register field at the high addresses of page 0 and can be changed by software. Priority levels range from 0 (low) to 7 (high) with NMI at level 8.

The on-chip interrupt controller decides whether an interrupt can be accepted by comparing its current priority with the interrupt mask level, and determines the order in which to accept competing interrupt requests. Interrupts that are not accepted immediately remain pending until they can be accepted later.

When the interrupt controller accepts an interrupt, it also decides whether to interrupt the CPU or to start the data transfer controller (DTC). This decision is controlled by the bits set in four data transfer enable registers (DTEA to DTED) in the register field. The DTC is started if the corresponding DTE bit is set to 1; otherwise a CPU interrupt is generated. DTC interrupts provide an efficient way to send and receive blocks of data via the serial communication interface, or to transfer data between memory and I/O without detailed CPU programming. DTC interrupts are described in section 6, Data Transfer Controller.

The hardware exception-handling sequence for a CPU interrupt clears the T bit in the status register to 0 and sets the interrupt mask level in bits I_2 to I_0 to the level of the interrupt it has accepted. This prevents the interrupt-handling routine from being interrupted except by a higher-level interrupt. The previous interrupt mask level is restored on the return from the interrupt-handling routine.

For further information on interrupts, see section 5, Interrupt Controller.

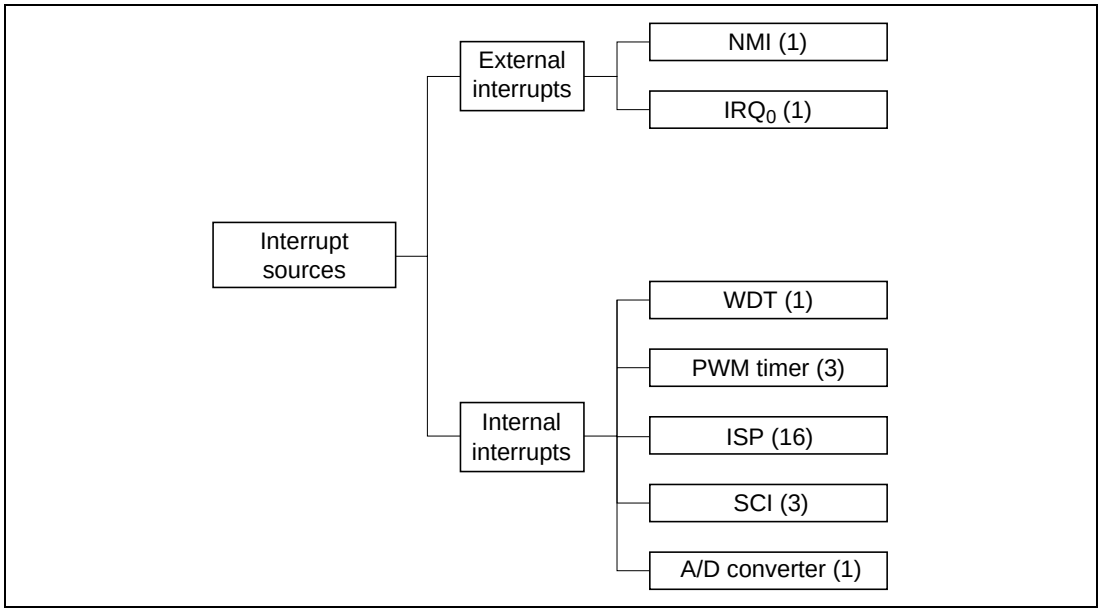


Figure 4.5 Interrupt Sources with Number of Interrupt Types

4.6 Invalid Instruction

An invalid instruction exception occurs if an attempt is made to execute an instruction with an undefined operation code or illegal addressing mode specification. The program counter value pushed onto the stack is the value of the program counter when the invalid instruction code was detected.

In the invalid-instruction exception-handling sequence the T bit of the status register is cleared to 0, but the interrupt mask level (I_2 to I_0) is not affected.

4.7 Trap Instructions and Zero Divide

A trap exception occurs when the TRAPA or TRAP/VS instruction is executed. A zero divide exception occurs if an attempt is made to execute a DIVXU instruction with a zero divisor.

In the exception-handling sequence for these exceptions the T bit of the status register is cleared to 0, but the interrupt mask level (I_2 to I_0) is not affected.

TRAPA Instruction: The TRAPA instruction always causes a trap exception. The TRAPA instruction includes a vector number from 0 to 15, allowing the user to provide up to sixteen different trap-handling routines.

TRAP/VS Instruction: When the TRAP/VS instruction is executed, a trap exception occurs if the overflow (V) bit in the condition code register is set to 1. If the V bit is cleared to 0, no exception occurs and the next instruction is executed.

DIVXU Instruction with Zero Divisor: An exception occurs if an attempt is made to divide by zero in a DIVXU instruction.

4.8 Cases in which Exception Handling is Deferred

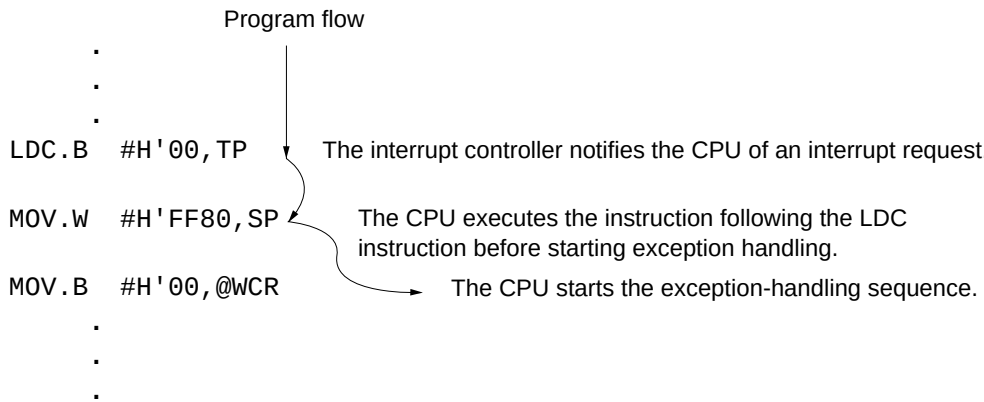
For the following three cases described in this subsection, the address error exception, trace exception, external interrupt (NMI, IRQ_0, and IRQ_1) requests, and internal interrupt requests (24 types) are not accepted immediately but are deferred until after the next instruction has been executed.

4.8.1 Instructions that Disable Interrupts

Interrupts are disabled immediately after the execution of the following five instructions: XORC, ORC, ANDC, LDC, and RTE.

Suppose that an internal interrupt is requested and the interrupt controller, after checking the interrupt priority and interrupt mask level, notifies the CPU of the interrupt, but the CPU is currently executing one of the five instructions listed above. After executing one of those instructions the CPU always proceeds to the next instruction. (And if the next instruction is again one of these five, the CPU also proceeds to the next instruction after that.) The exception-handling sequence starts after the next instruction, that is not one of these five, has been executed. The following is an example:

Example



4.8.2 Disabling of Exceptions Immediately after Reset

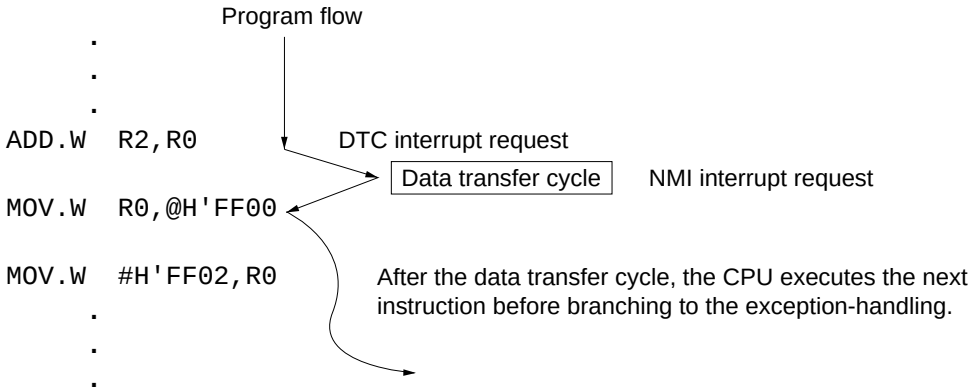
If an interrupt is accepted after reset and before the stack pointer (SP) is initialized, the program counter and status register will not be saved correctly, leading to a program crash. To prevent this, all interrupts including the NMI are disabled when the MCU comes out of the reset state, so the first instruction of the reset routine is always executed. As noted earlier, in the minimum mode, this first instruction should initialize the stack pointer (SP). In the maximum mode, the first instruction should be an LDC instruction that initializes the stack page register (TP) with the next instruction initializing the stack pointer.

4.8.3 Disabling of Interrupts after a Data Transfer Cycle

If an interrupt starts the data transfer controller and if another interrupt is requested during the data transfer cycle and when the data transfer cycle ends, the CPU always executes the next instruction before handling the second interrupt.

Even if a nonmaskable interrupt (NMI) occurs during a data transfer cycle, it is not accepted until the next instruction has been executed. See the following example.

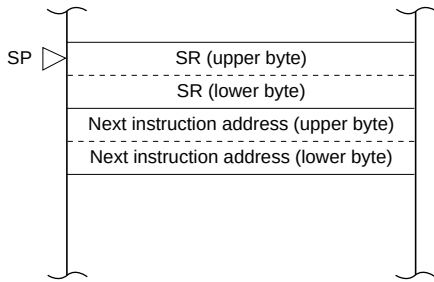
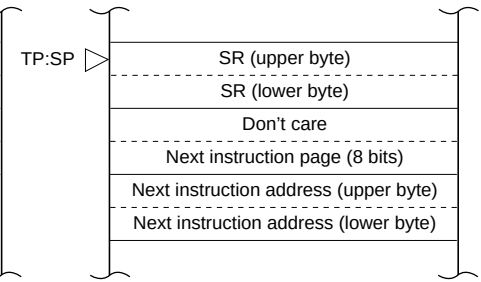
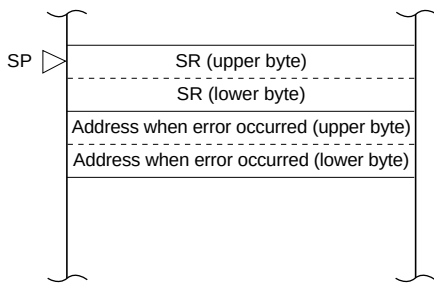
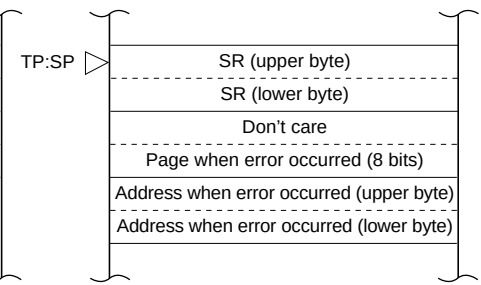
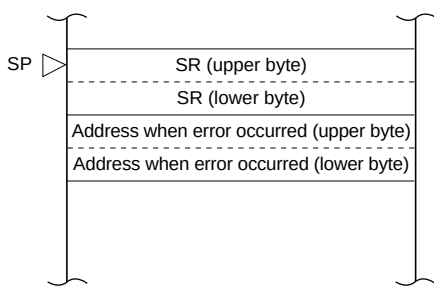
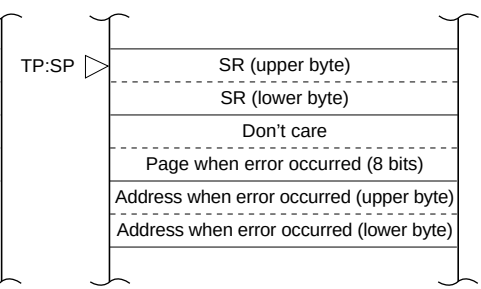
Example


Program flow
ADD.W R2,R0
MOV.W R0,@H'FF00
MOV.W #H'FF02,R0
DTC interrupt request
Data transfer cycle
NMI interrupt request
After the data transfer cycle, the CPU executes the next instruction before branching to the exception-handling.

4.9 Stack Status after Completion of Exception Handling

The status of the stack after an exception-handling sequence is performed depends on the various types of exceptions in the minimum and maximum modes (table 4.3).

Table 4.3 Stack After Exception Handling Sequence

Exception	Minimum Mode	Maximum Mode
Trace, interrupt, trap, zero divide (DIVXU)		
Note: The RTE instruction causes a return to the next instruction located after the instruction that was executed when the exception occurred.		
Invalid instruction		
Note: The program counter value pushed onto the stack is not necessarily the address of the first byte of the invalid instruction.		
Address error		
Note: The program counter value pushed onto the stack is the address of the next instruction located after the last instruction that was successfully executed.		

4.9.1 PC Value Pushed Onto Stack for Trace, Interrupts, Trap Instructions, and Zero Divide Exceptions

The program counter value pushed onto the stack for a trace, interrupt, trap, or zero divide exception is the address of the next instruction at the time when the interrupt was accepted. The RTE instruction accordingly causes a return to the next instruction loaded after the instruction that was executed before the exception-handling sequence.

4.9.2 PC Value Pushed Onto Stack for Address Error and Invalid Instruction Exceptions

The program counter value pushed onto the stack for an address error or invalid instruction exception differs depending on the conditions when the exception occurred.

4.10 Notes on Use of the Stack

If the stack pointer is set to an odd address, an address error will occur when the stack is accessed during interrupt handling or for a subroutine call. The stack pointer should always point to an even address. To keep the stack pointer pointing to an even address, a program should use word-size data as a standard when saving or restoring registers to and from the stack.

In the @–SP or @SP+ addressing mode, the CPU performs word access even if the instruction specifies byte size. (This is not true for the @–Rn and @Rn+ addressing modes when Rn is a register from R0 to R6.)

Section 5 Interrupt Controller

5.1 Overview

The interrupt controller resolves which interrupts to accept, and manages multiple interrupts. It also resolves on whether an interrupt should be served by the CPU or by the data transfer controller (DTC). This section explains the features of the interrupt controller, describes its internal structure and control registers, and details the handling of interrupts.

For detailed information on the data transfer controller, see section 6, Data Transfer Controller.

5.1.1 Features

- Interrupt priorities are user-programmable.

  User programs can set priority levels from 7 (high) to 0 (low) in four interrupt priority (IPR) registers for IRQ_0 and each on-chip supporting module—for every interrupt, except the nonmaskable interrupt (NMI). NMI has the highest priority level (8) and is always accepted. An interrupt with priority level 0 is always masked.
- Multiple interrupts on the same level are served in a default priority order.

  Lower priority interrupts remain pending until higher priority interrupts have been handled.
- For most interrupts, software can select whether to have the interrupt served by the CPU or the data transfer controller (DTC).

User programs can make this selection by setting and clearing bits in four data transfer enable (DTE) registers. The data transfer controller can be started by any interrupt except NMI, the error interrupt (ERI) from the serial communication interface, and the overflow interrupts (FOVI and OVI) from the timers.

5.1.2 Block Diagram

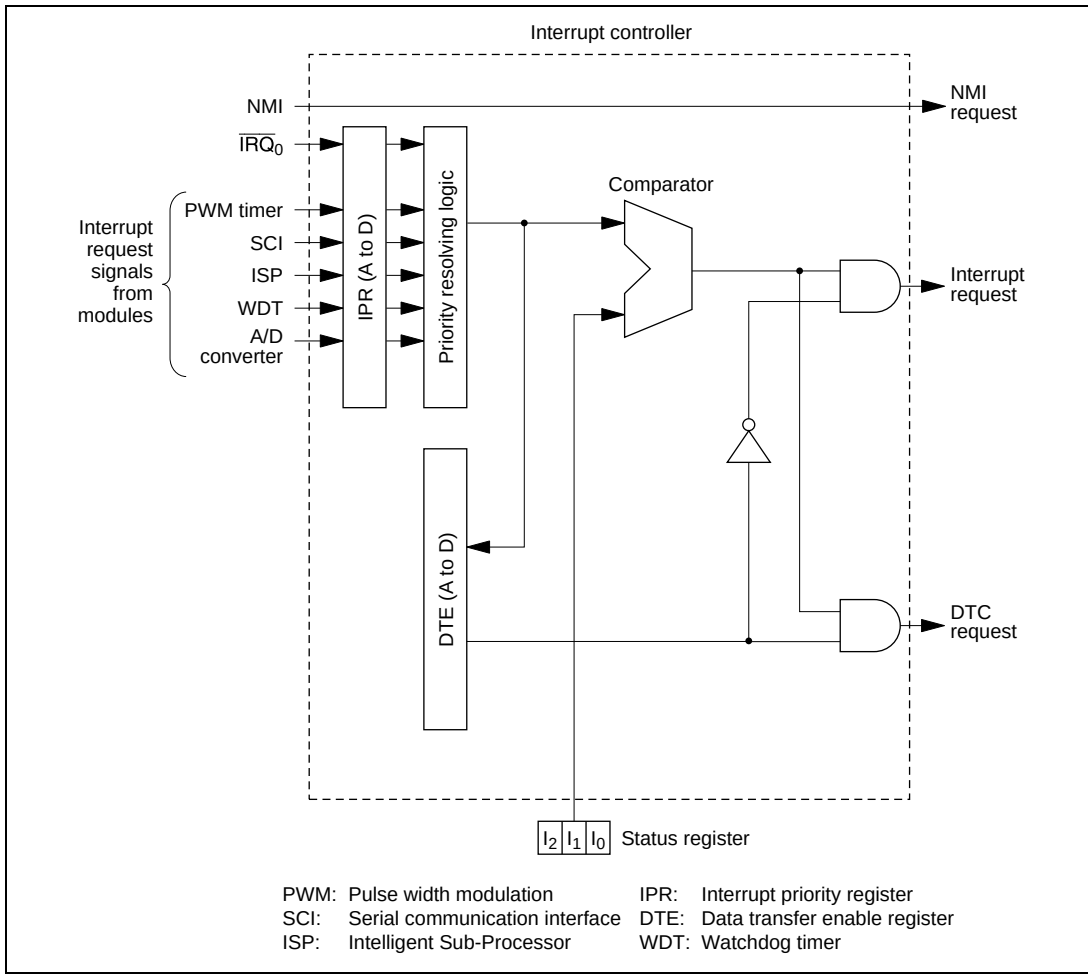


Figure 5.1 Interrupt Controller Block Diagram

5.1.3 Register Configuration

The four interrupt priority registers (IPRA to IPRD) and four data transfer enable registers (DTEA to DTED) are 8-bit registers located at addresses H'FF40 to H'FF47 in the register field in page 0 of the address space.

Table 5.1 Interrupt Controller Registers

Name	Abbreviation	Read/Write	Address	Initial Value
Interrupt priority registers A to D	IPRA	R/W	H'FF40	H'00
	IPRB	R/W	H'FF41	H'00
	IPRC	R/W	H'FF42	H'00
	IPRD	R/W	H'FF43	H'00
Data transfer enable registers A to D	DTEA	R/W	H'FF44	H'00
	DTEB	R/W	H'FF45	H'00
	DTEC	R/W	H'FF46	H'00
	DTED	R/W	H'FF47	H'00

5.2 Interrupt Types

There are 26 distinct types of interrupts: 2 external interrupts and 24 internal interrupts originating from the on-chip supporting modules.

5.2.1 External Interrupts

The two external interrupts are NMI and IRQ_0.

- NMI (Nonmaskable Interrupt): This interrupt has the highest priority level (8) and cannot be masked. An NMI is generated by an input to the NMI pin. The input at the NMI pin is edge-sensed. A user program can select whether to have the interrupt occur on the rising edge or falling edge of the NMI input by setting or clearing the nonmaskable interrupt edge bit (NMIEG) in system control register 1 (SYSCR1).

  In the NMI exception-handling sequence, the T (trace) bit in the CPU status register (SR) is cleared to 0, and the interrupt mask level in I_2 to I_0 is set to 7, masking all other interrupts.

- IRQ_0 (Interrupt Request 0): An IRQ_0 interrupt is requested by sinking the $\overline{IRQ_0}$ pin low. The input at the $\overline{IRQ_0}$ pin is level-sensed. Sinking $\overline{IRQ_0}$ low requests an IRQ_0 interrupt if the interrupt request enable 0 bit (IRQ_0E) in the SYSCR1 is set to 1.

  The IRQ_0 interrupt can be assigned any priority level from 7 to 0 by setting the corresponding value in the upper four bits of IPRA. If bit 4 of data transfer enable register A (DTEA) is set to

1, an IRQ_0 interrupt starts the data transfer controller. Otherwise, the interrupt is served by the CPU.

In the CPU interrupt-handling sequence for IRQ_0, the T bit of the status register is cleared to 0, and the interrupt mask level is set to the value in the upper four bits of IPRA.

5.2.2 Internal Interrupts

Twenty-four types of internal interrupts can be requested by the on-chip supporting modules. Each interrupt is separately vectored in the exception vector table, so it is not necessary for the user-coded interrupt handler routine to determine which type of interrupt has occurred.

An interrupt priority level from 7 to 0 can be assigned to each on-chip supporting module by setting interrupt priority registers A to D. Within each module, different interrupts have a fixed priority order. For most of these interrupts, values set in data transfer enable registers B to D can select whether to have the interrupt served by the CPU or the data transfer controller.

In the CPU interrupt-handling sequence, the T bit of the CPU status register is cleared to 0, and the interrupt mask level in bits I_2 to I_0 is set to the value in the IPR.

5.2.3 Interrupt Vector Table

For on-chip supporting modules, the priority level set in the interrupt priority register applies to the module as a whole: all interrupts from that module have the same priority level. A separate priority order is established among interrupts from the same module. See table 5.2. If the same priority level is assigned to two or more modules and two interrupts are requested simultaneously from these modules, they are served in the priority order indicated in the rightmost column in table 5.2.

Reset clears the interrupt priority registers so that all interrupts except NMI start with priority level 0, thus causing them to be unconditionally masked.

Table 5.2 Interrupts, Vectors, and Priorities

Interrupt		Assignable Priority Levels (Initial Level)	Priority IPR Bits	Within Module	Vector Table Entry Address		Priority Among Interrupts on Same Level*
					Minimum Mode	Maximum Mode	
NMI		8 (8)	—	—	H'16–H'17	H'2C–H'2F	High
IRQ_0, WDT		7 to 0 (0)	IPRA bits 6 to 4	1 0	H'40–H'41 H'44–H'45	H'80–H'83 H'88–H'8B	
PWM module	OCF0 OCF1 OCF2	7 to 0 (0)	IPRA bits 2 to 0	2 1 0	H'48–H'49 H'4A–H'4B H'4C–H'4D	H'90–H'93 H'94–H'97 H'98–H'9B	
ISP module	ISF0 ISF1 ISF2 ISF3	7 to 0 (0)	IPRB bits 2 to 0	3 2 1 0	H'50–H'51 H'52–H'53 H'54–H'55 H'56–H'57	H'A0–H'A3 H'A4–H'A7 H'A8–H'AB H'AC–H'AF	
	ISF4 ISF5 ISF6 ISF7	7 to 0 (0)	IPRB bits 6 to 4	3 2 1 0	H'58–H'59 H'5A–H'5B H'5C–H'5D H'5E–H'5F	H'B0–H'B3 H'B4–H'B7 H'B8–H'BB H'BC–H'BF	
	ISF8 ISF9 ISF10 ISF11	7 to 0 (0)	IPRC bits 2 to 0	3 2 1 0	H'60–H'61 H'62–H'63 H'64–H'65 H'66–H'67	H'C0–H'C3 H'C4–H'C7 H'C8–H'CB H'CC–H'CF	
	ISF12 ISF13 ISF14 ISF15	7 to 0 (0)	IPRC bits 4 to 0	3 2 1 0	H'68–H'69 H'6A–H'6B H'6C–H'6D H'6E–H'6F	H'D0–H'D3 H'D4–H'D7 H'D8–H'DB H'DC–H'DF	
SCI module	ERI RXI TXI	7 to 0 (0)	IPRD bits 6 to 4	2 1 0	H'70–H'71 H'72–H'73 H'74–H'75	H'E0–H'E3 H'E4–H'E7 H'E8–H'EB	
A/D converter module	ADI	7 to 0 (0)	IPRD bits 2 to 0	—	H'78–H'79	H'F0–H'F3	Low

Note: * If two or more interrupts are requested simultaneously, they are handled in the order of priority level as set in registers IPRA to IPRD. If they have the same priority level by being requested from the same on-chip supporting module, they are handled in a fixed priority order within the module. If they are requested from different modules to which the same priority level is assigned, they are handled in the order indicated in the rightmost column.

5.3 Register Descriptions

5.3.1 Interrupt Priority Registers A to D (IPRA to IPRD)

IRQ_0, IRQ_1, and the on-chip supporting modules are each assigned three bits in one of the four interrupt priority registers (IPRA to IPRD). These bits specify a priority level from 7 (high) to 0 (low) for interrupts from the corresponding source.

Bit	7	6	5	4	3	2	1	0
	—				—			
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R/W	R/W	R/W	R	R/W	R/W	R/W

Note: Bits 7 and 3 are reserved. They cannot be modified and are always read as 0.

Table 5.3 Assignment of Interrupt Priority Registers

	Interrupt Request Source	
Register	**Bits 6 to 4**	**Bits 2 to 0**
IPRA	IRQ_0, WDT	PWM
IPRB	ISP (IRQ_7–IRQ_4)	ISP (IRQ_3–IRQ_0)
IPRC	ISP (IRQ_{15}–IRQ_{12})	ISP (IRQ_{11}–IRQ_8)
IPRD	SCI	A/D converter

Each interrupt priority register specifies priority levels for two interrupt sources (table 5.3). A user program can assign the desired levels to these interrupt sources, for example, by writing 000 in bits 6 to 4 or bits 2 to 0 to set priority level 0, or 111 to set priority level 7.

Reset clears registers IPRA to IPRD to H'00, so all interrupts except NMI are initially masked.

When the interrupt controller receives one or more interrupt requests, it selects the request with the highest priority and compares its priority level with the interrupt mask level set in bits I_2 to I_0 in the CPU status register. If the priority level is higher than the mask level, the interrupt controller passes the interrupt request to the CPU (or starts the data transfer controller). If the priority level is lower than the mask level, the interrupt controller leaves the interrupt request pending until the level of the interrupt mask is lowered or the interrupt priority is raised. Similarly, if it receives two interrupt requests with the same priority level, the interrupt controller determines their priority by the order shown in table 5.2 and leaves the interrupt request with the lower priority pending.

5.3.2 Timing of Priority Setting

The interrupt controller requires two system clock (ø) periods to determine the priority level of an interrupt. Accordingly, when an instruction modifies an instruction priority register, the new priority does not take effect until after the next instruction has been executed.

5.4 Interrupt Handling Sequence

5.4.1 Interrupt Handling Flow

Note in figure 5.2 that the address error, trace exception, and NMI requests bypass the interrupt controller's priority resolver and are routed directly to the CPU.

1. Interrupt requests are generated by one or more on-chip supporting modules or external interrupt sources.
2. The interrupt controller checks the interrupt priorities set in IPRA to IPRD and selects the interrupt with the highest priority. Interrupts with lower priorities remain pending. Among interrupts with the same priority level, the interrupt controller determines the priority by a predetermined order (table 5.2).
3. The interrupt controller compares the priority level of the selected interrupt request with the mask level in the CPU status register (bits I_2 to I_0). If the priority level is equal to or less than the mask level, the interrupt request remains pending. If the priority level is higher than the mask level, the interrupt controller accepts the interrupt request and proceeds to the next step.
4. The interrupt controller checks the corresponding bit (if any) in the data transfer enable registers (DTEA to DTED). If this bit is set to 1, the data transfer controller is started. Otherwise, the CPU interrupt exception-handling sequence is started.

   If the data transfer enable bit is cleared to 0 or is nonexistent, the sequence proceeds as explained in the following steps. For the case in which the data transfer controller is started, see section 6, Data Transfer Controller.
5. After the CPU has completed executing the current instruction, the program counter and status register (in minimum mode), or program counter, code page register, and status register (in maximum mode) are saved to the stack (figure 5.3 (a) or (b)). The program counter value saved on the stack is the address of the next instruction to be executed.
6. The T (trace) bit of the status register is cleared to 0, and the priority level of the interrupt is copied to bits I_2 to I_0, thus masking any further interrupts unless they have a higher priority level.

   When an NMI is accepted, the interrupt mask level in bits I_2 to I_0 is set to 7.
7. The interrupt controller generates the vector address of the interrupt, and the entry at this address in the exception vector table is read to obtain the starting address of the user-coded interrupt handling routine.

In (7) the same difference between the minimum and maximum modes exists as in the reset handling sequence. In the minimum mode, one word is copied from the vector table to the program counter, then the interrupt-handling routine starts executing from the address indicated in the program counter. In the maximum mode, two words are read. The lower byte of the first word is copied to the code page register, and the second word is copied to the program counter. The interrupt-handling routine starts executing from the address indicated in the code page register and program counter.

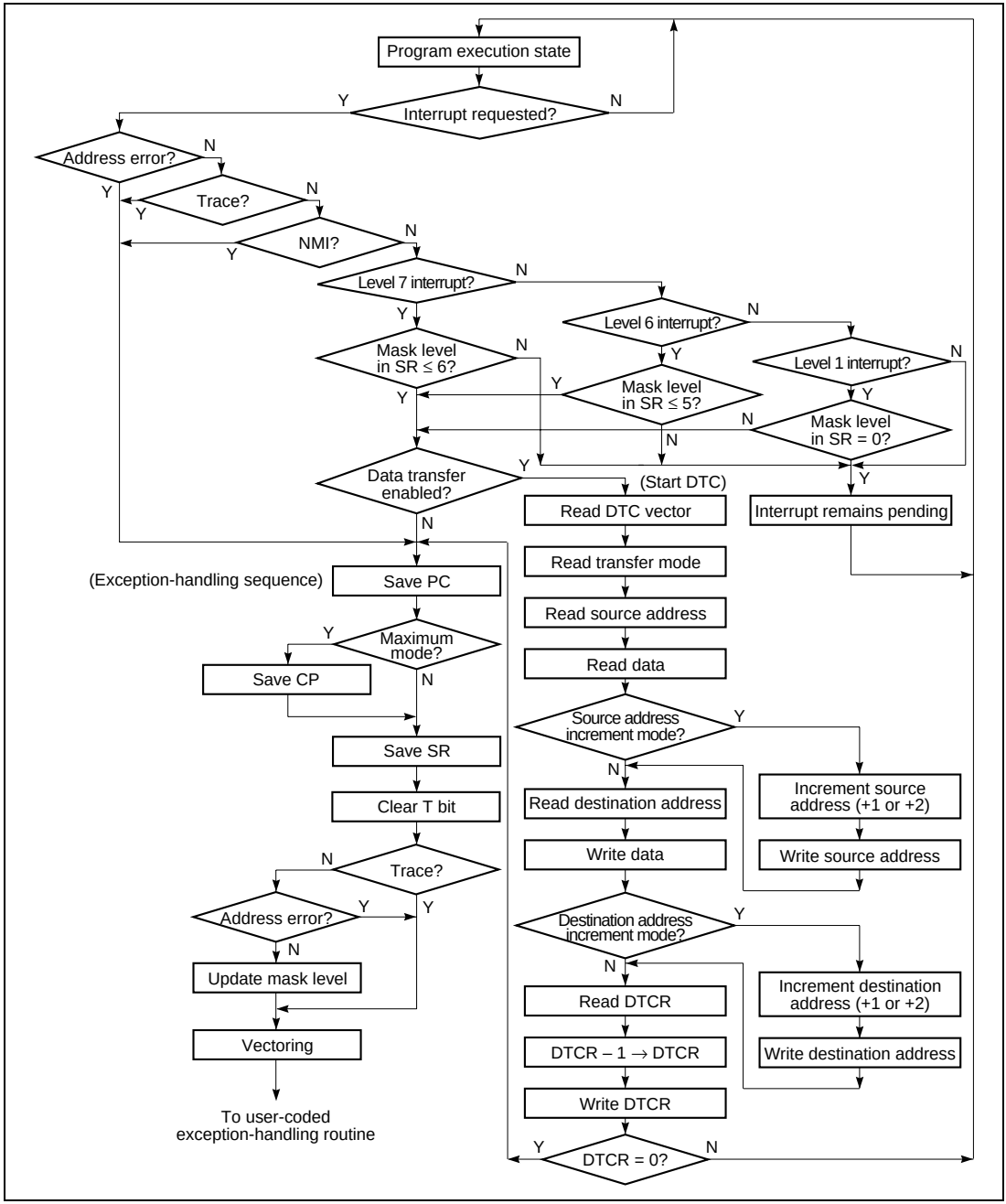
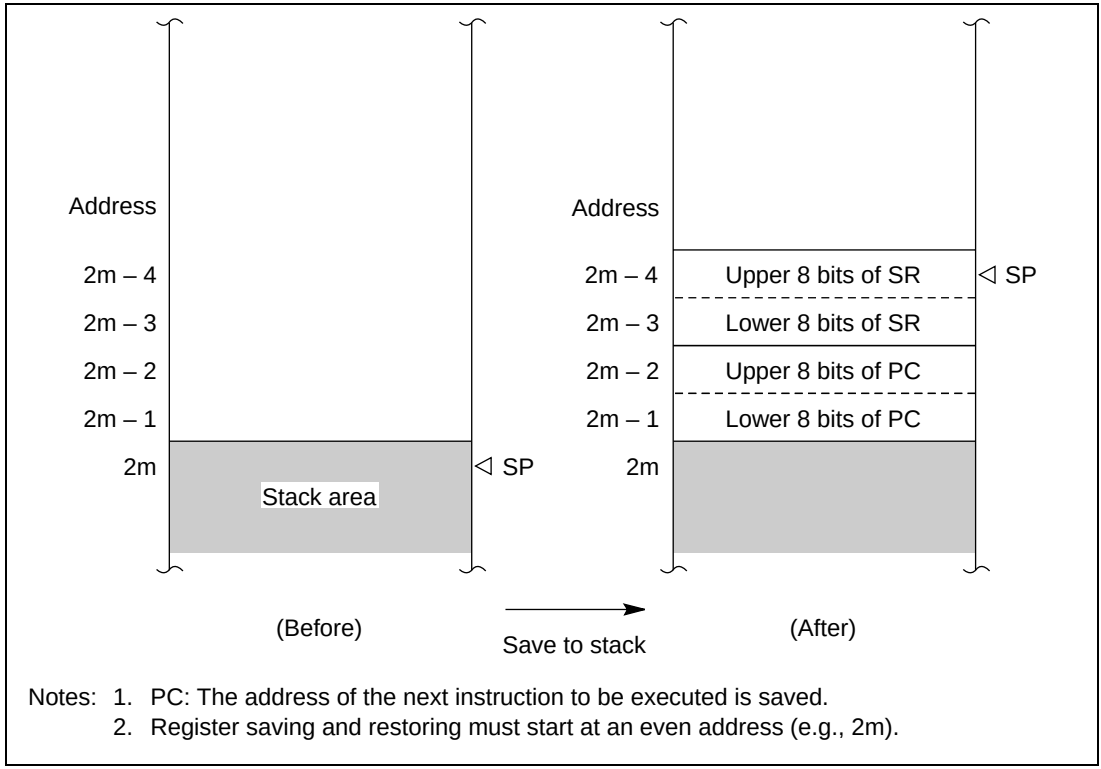


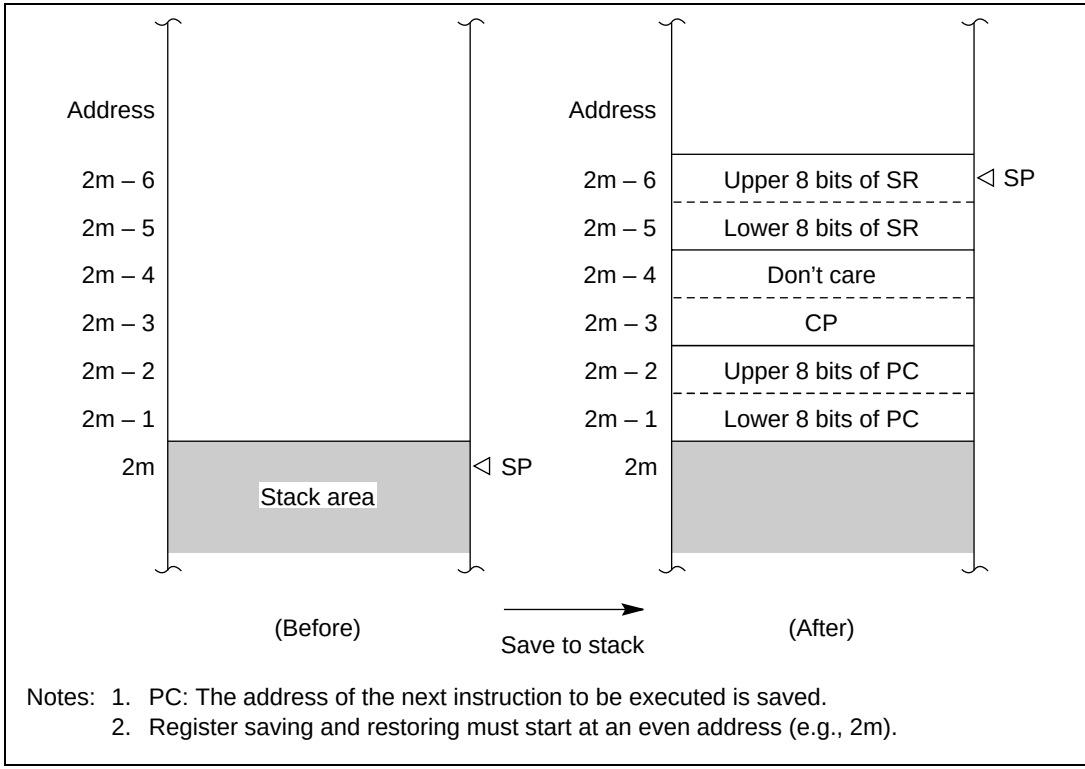
Figure 5.2 Interrupt Handling Flowchart

5.4.2 Stack Status After Interrupt Handling Sequence



Notes: 1. PC: The address of the next instruction to be executed is saved.
2. Register saving and restoring must start at an even address (e.g., 2m).

Figure 5.3 (a) Stack Before and After Interrupt Exception-Handling (Minimum Mode)



Notes: 1. PC: The address of the next instruction to be executed is saved.
2. Register saving and restoring must start at an even address (e.g., 2m).

Figure 5.3 (b) Stack Before and After Interrupt Exception-Handling (Maximum Mode)

5.4.3 Timing of Interrupt Exception-Handling Sequence

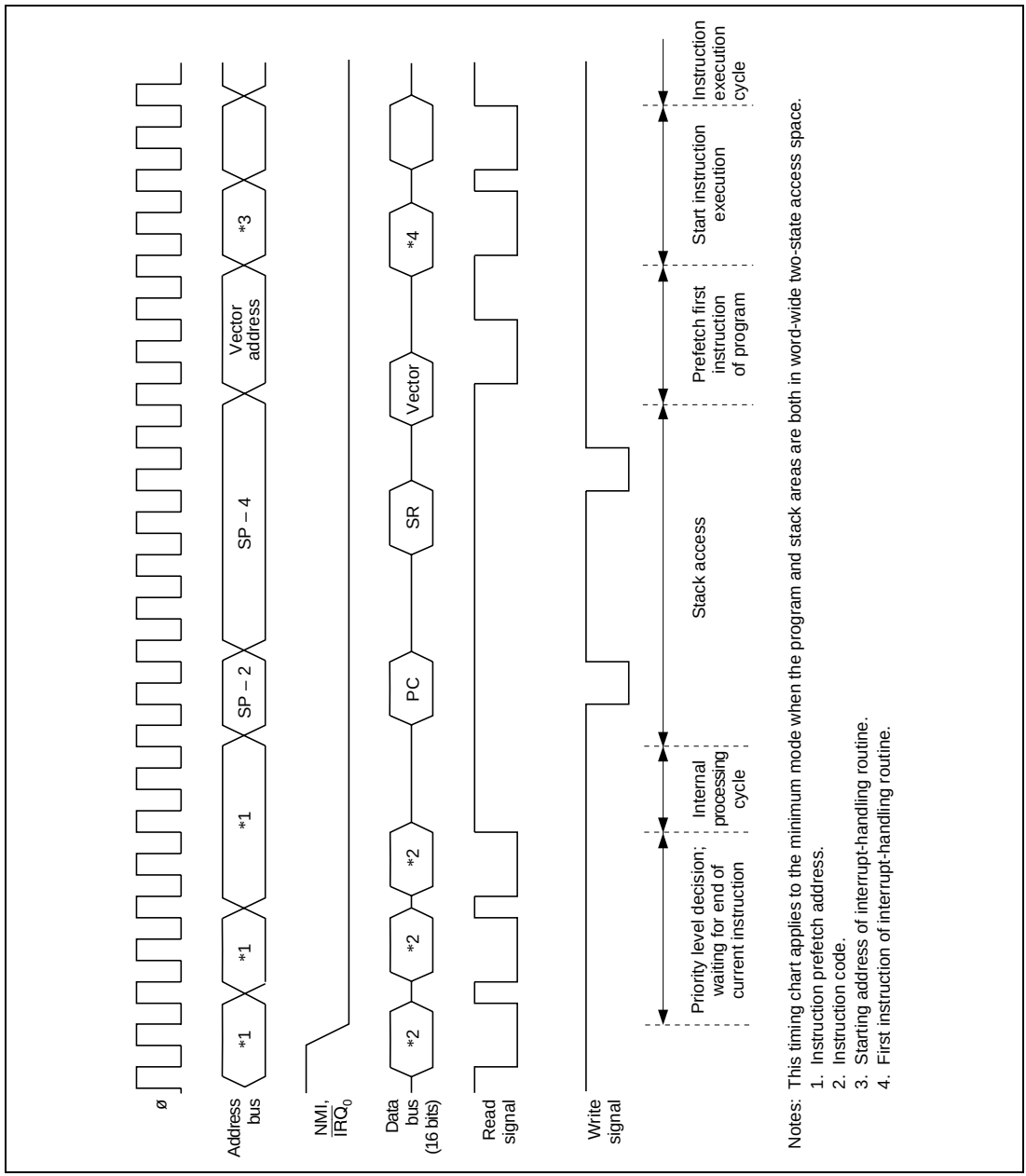
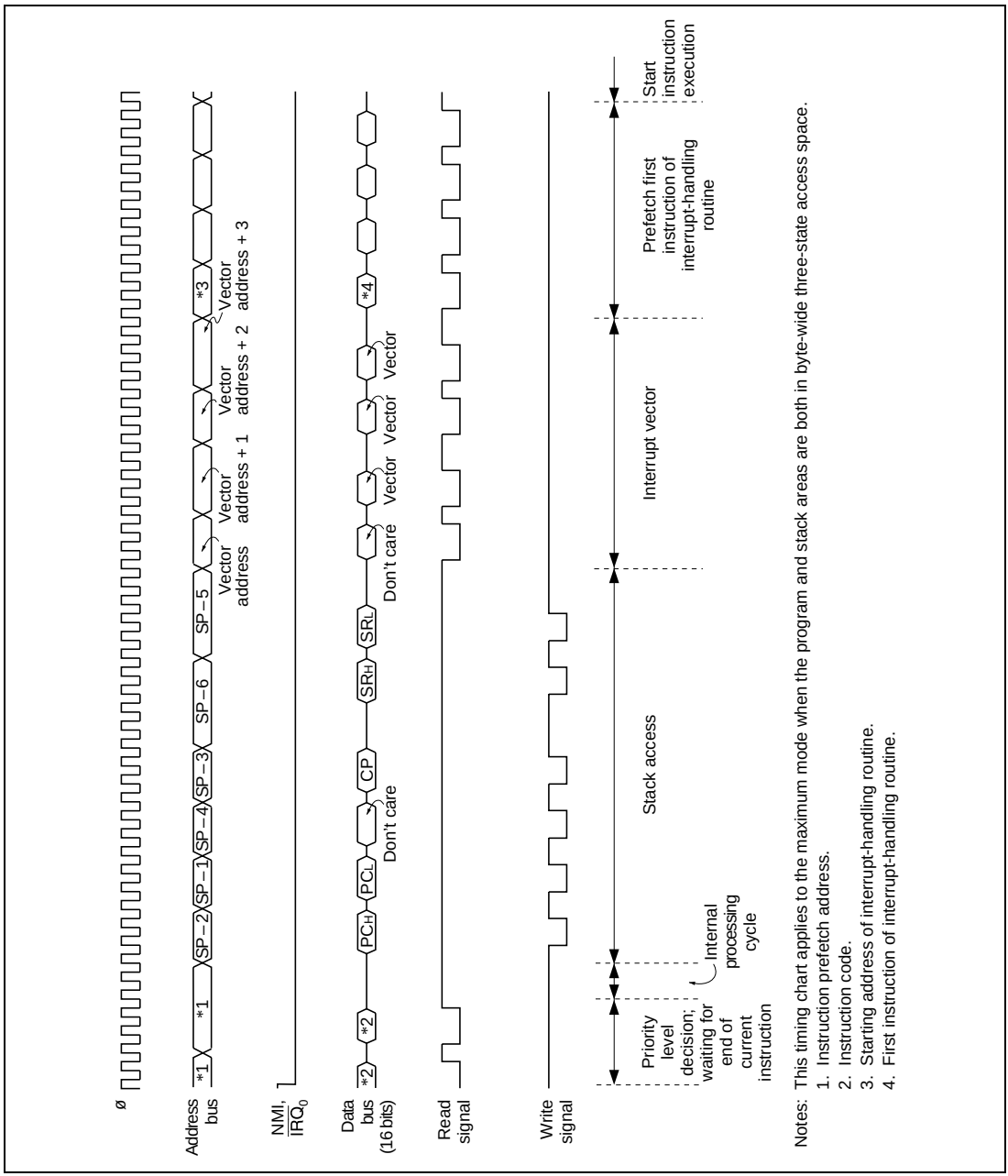


Figure 5.4 Interrupt Sequence (Minimum Mode)



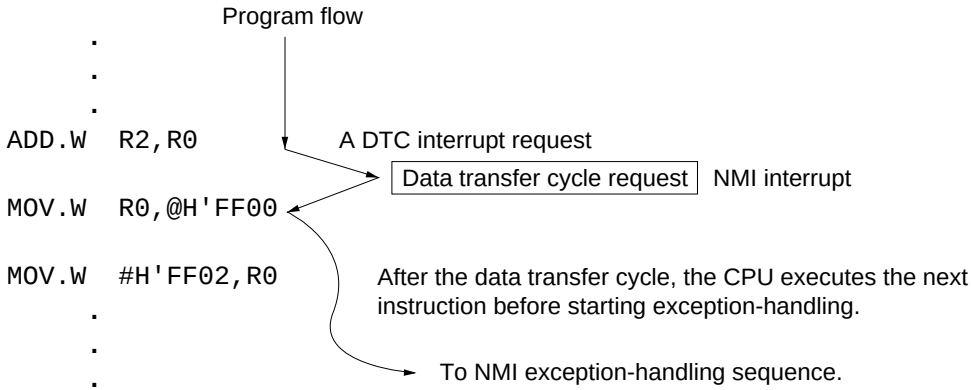
Notes: This timing chart applies to the maximum mode when the program and stack areas are both in byte-wide three-state access space.
1. Instruction prefetch address.
2. Instruction code.
3. Starting address of interrupt-handling routine.
4. First instruction of interrupt-handling routine.

Figure 5.5 Interrupt Sequence (Maximum Mode)

5.5 Interrupts During Operation of the Data Transfer Controller

If an interrupt is requested during a DTC data transfer cycle, the interrupt is not accepted until the data transfer cycle has been completed and the next instruction has been executed. This is true even if the interrupt is an NMI. An example is shown below.

Example



5.6 Interrupt Response Time

Depending on the source of the delay and mode, the number of states that may elapse between the generation of an interrupt request and the execution of the first instruction of the interrupt-handling routine is determined (table 5.4). Since word access is performed to memory, the fastest interrupt service can be achieved by placing the program into RAM.

Table 5.4 Number of States Before Interrupt Service

		Number of States	
Cause of Delay		**Minimum Mode**	**Maximum Mode**
Interrupt priority resolving and comparison with mask level in CPU status register		2 states	2 states
Maximum number of states up to completion specifying all of current instruction	Instruction is in word-wide two-state access space	x (x = 38 for LDM instruction registers)	x
	Instruction is in byte-wide three-state access space	y (y = 74 + 16m for LDM instruction specifying all registers)	y
Number of states from saving PC and SR or PC, CP, and SR up until instruction prefetch	Stack is in word-wide two-state access space	16	21
	Stack is in byte-wide three-state access space	28 + 6m	41 + 10m
Stack is in word-wide two-state access space	Instruction is in word-wide two-state access space	18 + x (56)	23 + x (61)
	Instruction is in byte-wide three-state access space	18 + y (92 + 16m)	23 + y (97 + 16m)
Stack is in byte-wide three-state access space	Instruction is in word-wide two-state access space	30 + 6m + x (68 + 6m)	43 + 10m + x (81 + 10m)
	Instruction is in byte-wide three-state access space	30 + 6m + y (104 + 22m)	43 + 10m + y (117 + 26m)

Notes: m: Number of wait states inserted during external memory access.
Values in parentheses are for the LDM instruction.

Section 6 Data Transfer Controller

6.1 Overview

The H8/570 microcontroller includes a data transfer controller (DTC) that can be enabled by designated interrupts to transfer data from a source address to a destination address located in page 0. These addresses include, in particular, the registers of the on-chip supporting modules and I/O ports. Typically, the DTC can be used to change the setting of a control register of an on-chip supporting module in response to an interrupt from that module, or to transfer data from memory to an I/O port or the serial communication interface. Once the DTC is set up, the transfer is interrupt-driven, so it proceeds independently of program execution, although program execution temporarily stops while each byte or word is being transferred.

6.1.1 Features

- The source address and destination address can be set anywhere in the 64-kbyte address space of page 0
- The DTC can be programmed to transfer one byte or one word of data per interrupt
- The DTC can be programmed to increment the source address and/or destination address after each byte or word is transferred
- After transferring a designated number of bytes or words, the DTC generates a CPU interrupt with the vector of the interrupt source that started the DTC
- The designated data transfer count can be set from 1 to 65,536 bytes or words

6.1.2 Block Diagram

The four DTC control registers (DTMR, DTSR, DTDR, and DTCR) are transparent to the CPU, but corresponding information is kept in a register information table in memory. A separate table is maintained for each DTC interrupt type. When an interrupt requests DTC service, the DTC loads its control registers from the table in memory, transfers the byte or word of data, and writes any altered register information back to memory.

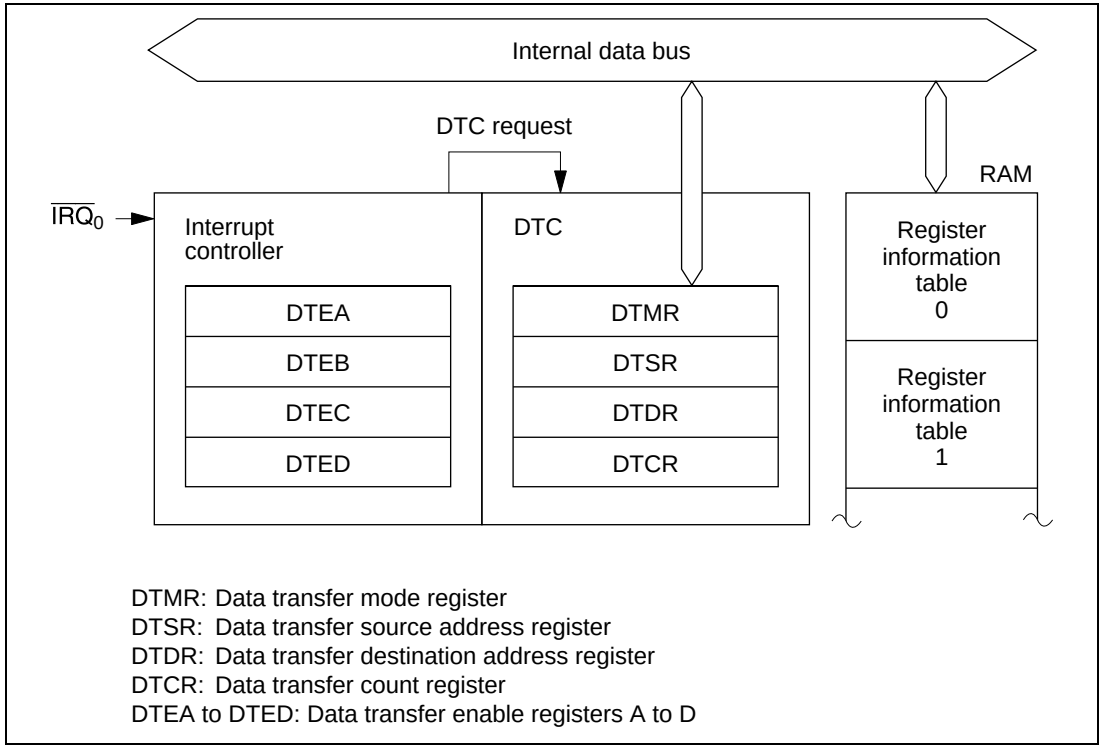


Figure 6.1 Block Diagram of Data Transfer Controller

6.1.3 Register Configuration

The four DTC control registers (table 6.1) are not located in the address space and cannot be written or read by the CPU. To set information in these registers, a program must write the information in a table in memory from which it will be loaded by the DTC.

Table 6.1 Internal Control Registers of the DTC

Register	Abbreviation	Read/Write
Data transfer mode register	DTMR	Disabled
Data transfer source address register	DTSR	Disabled
Data transfer destination address register	DTDR	Disabled
Data transfer count register	DTCR	Disabled

Enabling the DTC is done by the four data transfer enable registers (table 6.2), which are located in the high addresses of page 0.

Table 6.2 Data Transfer Enable Registers

Register	Abbreviation	Read/Write	Address	Initial Value
Data transfer enable registers A to D	DTEA	R/W	H'FF44	H'00
	DTEB	R/W	H'FF45	H'00
	DTEC	R/W	H'FF46	H'00
	DTED	R/W	H'FF47	H'00

6.2 Register Descriptions

6.2.1 Data Transfer Mode Register (DTMR)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Sz	SI	DI	—	—	—	—	—	—	—	—	—	—	—	—	—
Read/Write	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

The data transfer mode register is a 16-bit register with the first three bits designating the data size and specifying whether to increment the source and destination addresses.

Bit 15—Sz (Size): This bit designates the size of the data transferred.

Bit 15 Sz	Description
0	Byte transfer
1	Word transfer*

Note: * For word transfer, the source and destination addresses must be even addresses.

Bit 14—SI (Source Increment): This bit specifies whether to increment the source address.

Bit 14 SI	Description
0	Source address is not incremented.
1	If Sz = 0, source address is incremented by 1 after each data transfer. If Sz = 1, source address is incremented by 2 after each data transfer.

Bit 13—DI (Destination Increment): This bit specifies whether to increment the destination address.

Bit 13 DI	Description
0	Destination address is not incremented.
1	If Sz = 0, destination address is incremented by 1 after each data transfer. If Sz = 1, destination address is incremented by 2 after each data transfer.

Bits 12 to 0—Reserved Bits

6.2.2 Data Transfer Source Address Register (DTSR)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read/Write	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

The data transfer source register is a 16-bit register that designates the data transfer source address. For word transfer an even address must be specified. In the maximum mode, this address is implicitly located in page 0.

6.2.3 Data Transfer Destination Register (DTDR)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read/Write	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

The data transfer destination register is a 16-bit register that designates the data transfer destination address. For word transfer an even address must be specified. In the maximum mode, this address is implicitly located in page 0.

6.2.4 Data Transfer Count Register (DTCR)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read/Write	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

The data transfer count register is a 16-bit register that counts the number of bytes or words of data remaining to be transferred. The initial count can be set from 1 to 65,536. A register value

of 0 designates an initial count of 65,536.

The data transfer count register is decremented automatically after each byte or word is transferred. When its value reaches 0, indicating that the designated number of bytes or words have been transferred, a CPU interrupt is generated with the vector of the interrupt that requested the data transfer.

6.2.5 Data Transfer Enable Registers A to D (DTEA to DTED)

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

These four registers designate whether an interrupt starts the DTC. The bits in these registers are assigned to interrupts (table 6.3). No bits are assigned to the NMI, WDT, ISF3, ISF7, ISF11, and ERI interrupts, which cannot request data transfers.

Table 6.3 Assignment of Data Transfer Enable Registers to Interrupt Sources

Register	Interrupt Source	Bits 7 to 4				Interrupt Source	Bits 3 to 0			
		7	6	5	4		3	2	1	0
DTEA	IRQ_0	—	—	IRQ_0	—	PWM	—	OCF2	OCF1	OCF0
DTEB	ISP	—	ISF6	ISF5	ISF4	ISP	—	ISF2	ISF1	ISF0
DTEC	ISP	—	ISF14	ISF13	ISF12	ISP	—	ISF10	ISF9	ISF8
DTED	SCI	—	TXI	RXI	—	A/D converter	—	—	—	ADI

Note: Bits marked by — should always be cleared to 0.

If the bit for a certain interrupt is set to 1, that interrupt is regarded as a request for the DTC. If the bit is cleared to 0, the interrupt is regarded as a CPU interrupt request.

Only the 19 interrupts listed in table 6.3 can request the DTC. DTE bits not assigned to any interrupt (indicated by — in table 6.3) should be cleared to 0.

Note on Timing of DTE Modifications: The interrupt controller requires two system clock (ø) periods to determine the priority level of an interrupt. Accordingly, when an instruction modifies a data transfer enable register, the new setting does not take effect until after the next instruction has been executed.

6.3 Data Transfer Operation

6.3.1 Data Transfer Cycle

When enabled by an interrupt, the DTC executes the following data transfer cycle (figure 6.2):

1. From the DTC vector table, the DTC reads the address at which the register information table for that interrupt is located in memory.
2. The DTC loads the data transfer mode register and source address register from this table and reads the data (one byte or word) from the source address.
3. If so specified in the mode register, the DTC increments the source address register and writes the new source address back to the table in memory.
4. The DTC loads the data transfer destination address register and writes the byte or word of data to the destination address.
5. If so specified in the mode register, the DTC increments the destination address register and writes the new destination address back to the table in memory.
6. The DTC loads the data transfer count register from the table in memory, decrements the data count, and writes the new count back to memory.
7. If the data transfer count is now 0, the DTC generates a CPU interrupt. The interrupt vector is the vector of the interrupt type that enabled the DTC.

At an appropriate point during this procedure the DTC also clears the interrupt request by clearing the corresponding flag bit in the status register of the on-chip supporting module to 0. (For IRQ_0 or IRQ_1, the DTC clears an internal latch.)

However, the DTC does not clear the data transfer enable bit in the data transfer enable register. This action, if necessary, must be taken by the user-coded interrupt-handling routine invoked at the end of the transfer.

For the procedure from the occurrence of the interrupt up to the start of the data transfer cycle, see section 5.4.1, Interrupt Handling Flow.

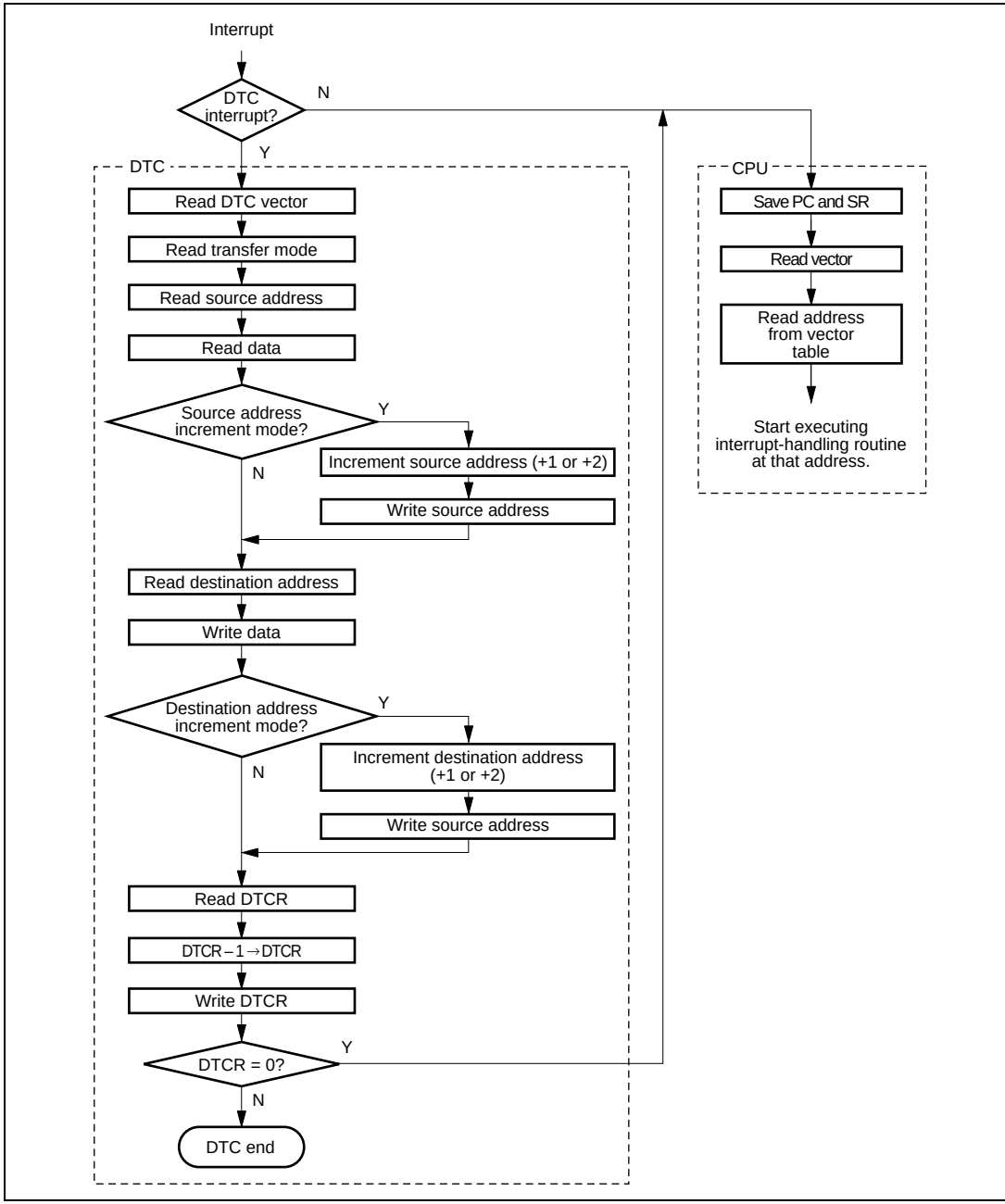
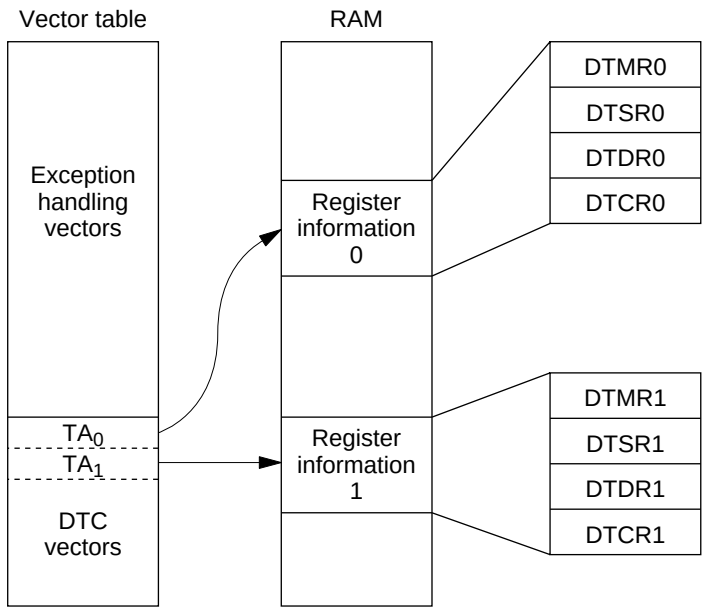


Figure 6.2 Flowchart of Data Transfer Cycle

6.3.2 DTC Vector Table

The DTC vector table is located immediately following the exception vector table at the beginning of page 0 in memory. For each interrupt that can request the DTC, the DTC vector table provides a pointer to an address in memory where the table of DTC control register information for that interrupt is stored. The register information tables can be placed in any available location in page 0.



Note: TA0, TA1,...TAn: Addresses of DTC register information tables in memory.

Normally, the register information tables are placed in RAM. If the software does not need to modify the register information (addresses are fixed and transfer count is 1), it can be placed on ROM.

Figure 6.3 DTC Vector Table

In minimum mode, each entry in the DTC vector table consists of two bytes, pointing to an address in page 0. In maximum mode, for compatibility reasons, each DTC vector table entry consists of four bytes but the first two bytes are ignored; the last two bytes point to an address which is implicitly assumed to be in page 0, regardless of the current page specifications. See figure 6.4.

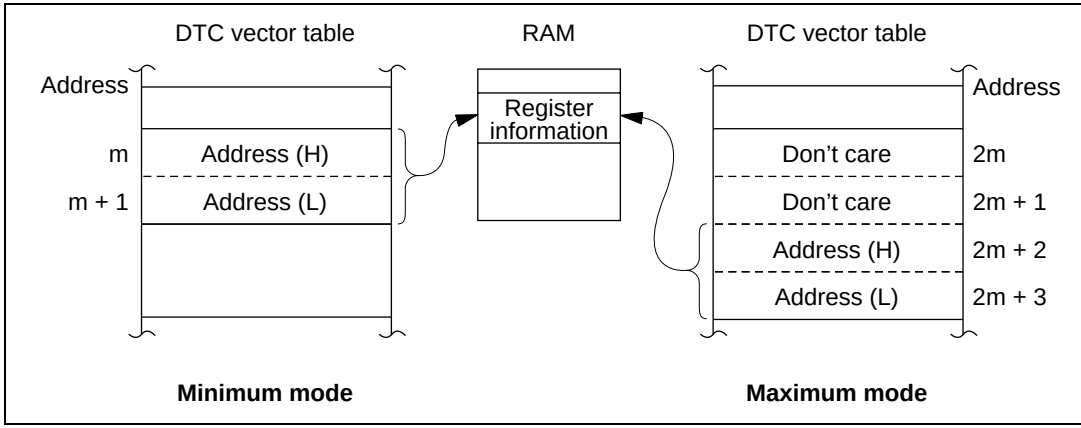


Figure 6.4 DTC Vector Table Entry

Table 6.4 Addresses of DTC Vectors

		Address of DTC Vector	
Interrupt		**Minimum Mode**	**Maximum Mode**
IRQ_0		H'0080–H'0081	H'0100–H'0103
PWM timer	OCF0	H'0088–H'0089	H'0110–H'0113
	OCF1	H'008A–H'008B	H'0114–H'0117
	OCF2	H'008C–H'008D	H'0118–H'011B
ISP	ISF0	H'0090–H'0091	H'0120–H'0123
	ISF1	H'0092–H'0093	H'0124–H'0127
	ISF2	H'0094–H'0095	H'0128–H'012B
	ISF4	H'0098–H'0099	H'0130–H'0133
	ISF5	H'009A–H'009B	H'0134–H'0137
	ISF6	H'009C–H'009D	H'0138–H'013B
	ISF8	H'00A0–H'00A1	H'0140–H'0143
	ISF9	H'00A2–H'00A3	H'0144–H'0147
	ISF10	H'00A4–H'00A5	H'0148–H'014B
	ISF12	H'00A8–H'00A9	H'0150–H'0153
	ISF13	H'00AA–H'00AB	H'0154–H'0157
	ISF14	H'00AC–H'00AD	H'0158–H'015B
SCI	RXI	H'00B2–H'00B3	H'0164–H'0167
	TXI	H'00B4–H'00B5	H'0168–H'016B
A/D converter	ADI	H'00B8–H'00B9	H'0170–H'0173

6.3.3 Location of Register Information in Memory

For each interrupt, the DTC control register information is stored in four consecutive words in memory (figure 6.5).

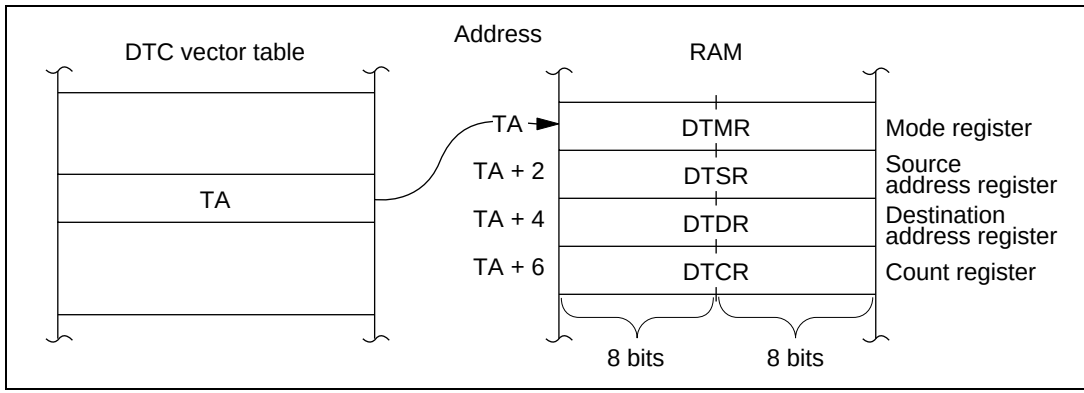


Figure 6.5 Order of Register Information

6.3.4 Length of Data Transfer Cycle

Assuming that the DTC control register information is stored in RAM, a certain number of states are required for loading and saving the DTC control registers and for transferring one byte or word of data. Two cases are considered: a transfer between RAM and an I/O port register or on-chip supporting module (i.e., a register in the register field from addresses H'FE80 to H'FFFF); and a transfer between such a register and external RAM.

Table 6.5 Number of States Per Data Transfer

Increment Mode		Word-Wide Two State Access Space ↔ Module or I/O Register		Byte-Wide Three-State Access Space ↔ Module or I/O Register	
Source (SI)	Destination (DI)	Byte Transfer	Word Transfer	Byte Transfer	Word Transfer
0	0	31	34	32	38
0	1	33	36	34	40
1	0	33	36	34	40
1	1	35	38	36	42

Note: Units are in number of states.

The values in table 6.5 are calculated from the formula:

$$N = 26 + 2 \times SI + 2 \times DI + M_S + M_D$$

M_S: Number of states for reading source data
M_D: Number of states for writing destination data

The values of M_S and M_D depend on the data location as follows:

Byte or word data in RAM: 2 states
Byte data in external RAM or register field: 3 states
Word data in external RAM or register field: 6 states

If the DTC control register information is stored in byte-wide three-state access space, $20 + 4 \times SI + 4 \times DI$ must be added to the values in table 6.5.

The values given above do not include the time between the occurrence of the interrupt request and the enabling of the DTC. This period includes two states for the interrupt controller to check the priority and a variable wait period until the end of the current CPU instruction. The maximum number of states this period will be is the sum of the values listed in table 6.6, excluding the last cause of delay.

If the data transfer count is 0 at the end of a data transfer cycle, the number of states, starting from the end of the data transfer cycle up until the first instruction of the user-coded interrupt-handling routine is executed, is the value given for the last cause of delay in table 6.6.

Table 6.6 Number of States Before Interrupt Service

		Number of States	
Cause of Delay		**Minimum Mode**	**Maximum Mode**
Interrupt priority resolving and comparison with mask level in CPU status register		2 states	2 states
Maximum number of states up to completion of current instruction	Instruction is in word-wide two-state access space	38	38
		(LDM instruction specifying all registers)	
	Instruction is in byte-wide three-state access space	74 + 16m	74 + 16m
		(LDM instruction specifying all registers)	
Number of states from saving of PC and SR, or PC, CP, and SR up until instruction prefetch	Stack is in word-wide two-state access space	16	21
	Stack is in byte-wide three-state access space	28 + 6m	41 + 10m

Note: m: Number of wait states inserted during external memory access.

6.4 Procedure for Using the DTC

The program that controls the DTC to transfer data must do the following:

1. Set the appropriate DTMR, DTSR, DTDR, and DTCR register information in the memory location indicated in the DTC vector table.
2. Set the data transfer enable bit of the pertinent interrupt to 1, and set the priority of the interrupt source (in the interrupt priority register) and the interrupt mask level (in the CPU status register) so that the interrupt can be accepted.
3. Set the interrupt enable bit in the control register for the interrupt source. (For IRQ_0 and IRQ_1, the control register is system control register 1, SYSCR1).

Following these preparations, the DTC will be started each time an interrupt occurs. When the number of bytes or words designated by the DTCR value have been transferred, after transferring the last byte or word, the DTC generates a CPU interrupt.

The user-coded interrupt-handling routine must be able to prepare for or disable further DTC data transfer (i.e., by readjusting the data transfer count or clearing the interrupt enable bit). If no action is taken by the user-coded interrupt-handling routine, the next interrupt of the same type will start the DTC with an initial data transfer count of 65,536.

6.5 Example of Using the DTC

Purpose: To receive 128 bytes of serial data via the serial communication interface.

Conditions:

- Operating mode: Minimum mode
- Received data are to be stored in consecutive addresses starting at H'FC00
- DTC control register information for the RXI interrupt is stored at addresses H'FB80 to H'FB87
- Accordingly, the DTC vector table contains H'FB at address H'00AA and H'80 at address H'00AB
- The desired interrupt mask level in the CPU status register is 4, and the desired SCI interrupt priority level is 5

Procedure:

1. The user program sets DTC control register information in addresses H'FB80 to H'FB87 as shown (table 6.7).

Table 6.7 DTC Control Register Information Set in RAM

Address	Register	Description	Value Set
H'FB80	DTMR	Byte transfer Source address fixed Increment destination address	H'2000
H'FB82	DTSR	Address of SCI receive data register	H'FE9D
H'FB84	DTDR	Address H'FC00	H'FC00
H'FB86	DTCR	Number of bytes to be received: 128	H'0080

2. The program sets the RI (SCI receive interrupt) bit in the data transfer enable register (bit 5 of register DTED) to 1.
3. The program sets the interrupt mask in the CPU status register to 4, and sets the SCI interrupt priority in bits 6 to 4 of interrupt priority register IPRD to 5.
4. The program sets the SCI to the appropriate receive mode, and sets the receive interrupt enable (RIE) bit in the serial control register (SCR) to 1 to enable receive interrupts.
5. Thereafter, each time the SCI receives one byte of data, it requests an RXI interrupt, which the interrupt controller directs toward the DTC. The DTC then transfers the byte from the SCI receive data register (RDR) into RAM (figure 6.6), and clears the interrupt request before ending.
6. When 128 bytes have been transferred (DTCR = 0), the DTC generates a CPU interrupt. The interrupt type is RXI.
7. The user-coded RXI interrupt-handling routine processes the received data and disables further data transfer (by clearing the RIE bit).

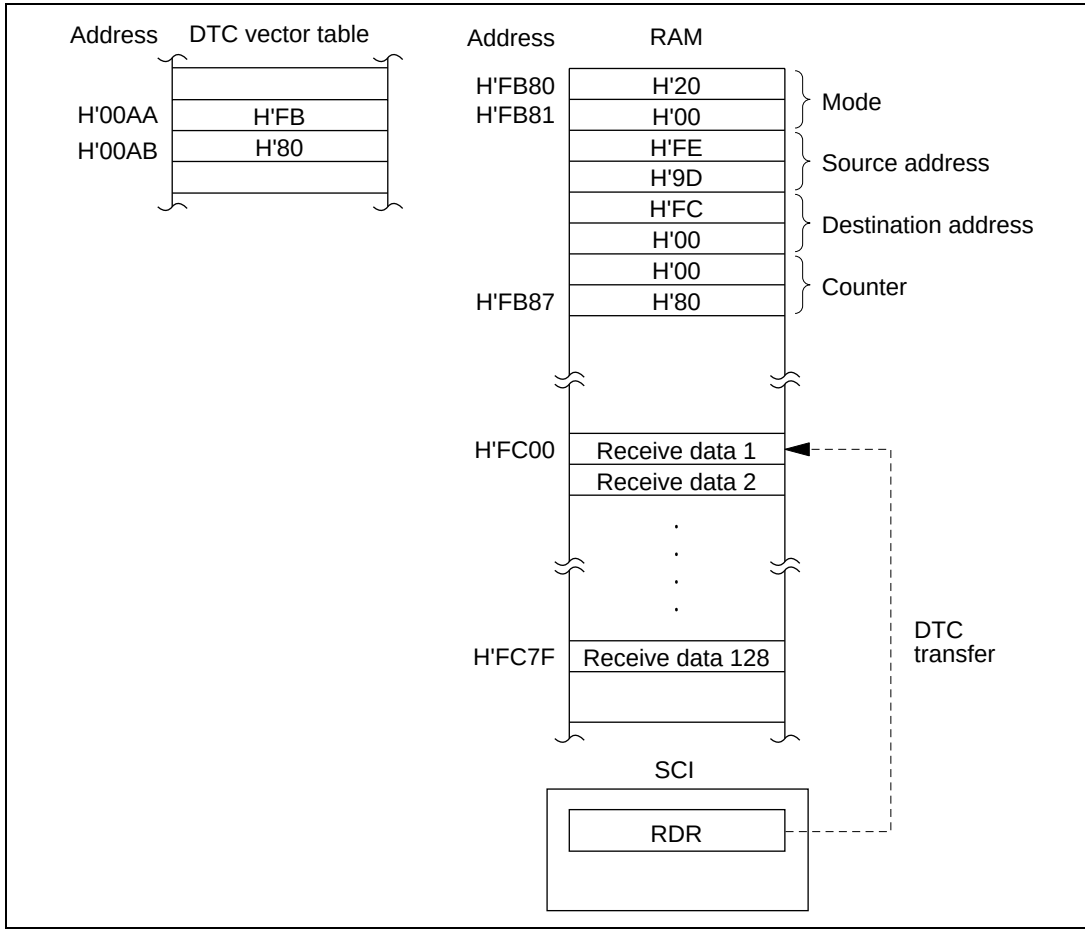


Figure 6.6 Use of DTC to Receive Data via Serial Communication Interface

Section 7 Wait-State Controller

7.1 Overview

To simplify interfacing to low-speed external devices, the H8/570 has an on-chip wait-state controller (WSC) that can insert wait states (T_w) to prolong bus cycles.

The wait-state function can be used both in CPU and DTC or ISP access cycles for external addresses. It can not be used in access cycles for on-chip supporting modules or internal memory. The T_w states are inserted between the T_2 state and T_3 state in the bus cycle. The number of wait states can be selected by a value set in the wait-state control register (WCR), or by asserting the $\overline{\text{WAIT}}$ pin low for the required interval.

7.1.1 Features

- Selection of three operating modes: Programmable wait mode, pin wait mode, or pin auto-wait mode
- 0, 1, 2, or 3 wait states can be inserted. For the pin wait mode, 4 or more states can be inserted by asserting the $\overline{\text{WAIT}}$ pin low.

7.1.2 Block Diagram

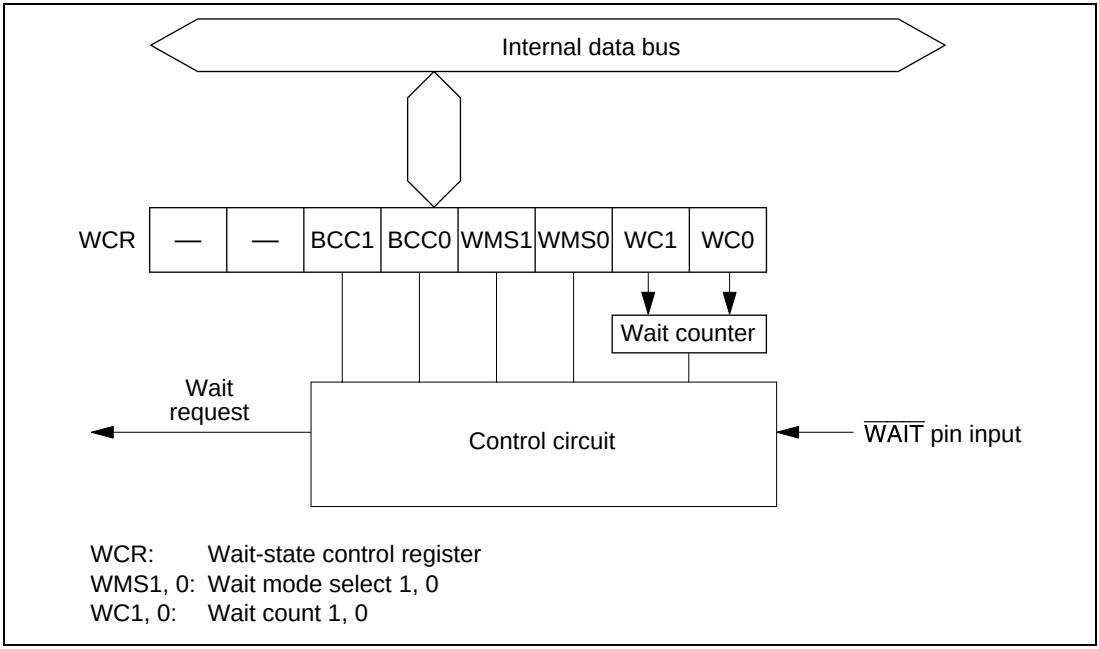


Figure 7.1 Block Diagram of Wait-State Controller

7.1.3 Register Configuration

The wait-state controller has one control register, the wait-state control register.

Name	Abbreviation	Read/Write	Initial Value	Address
Wait-state control register	WCR	R/W	H'C3	H'FF48

7.2 Wait-State Control Register

The wait-state control register (WCR) is an 8-bit register that specifies the wait mode and the number of wait states to be inserted. Reset initializes the WCR to specify the programmable wait mode with three wait states. The WCR is not initialized in the software standby mode.

Bit	7	6	5	4	3	2	1	0
	—	—	BCC1	BCC0	WMS1	WMS0	WC1	WC0
Initial value	1	1	0	0	0	0	1	1
Read/Write	—	—	R/W	R/W	R/W	R/W	R/W	R/W

Bits 7 and 6—Reserved: These bits cannot be modified and are always read as 1.

Bits 5 and 4—Bus Cycle Control 1 and 0 (BCC1 and BCC0): During the 16-bit data bus mode, these bits specify the number of bus cycle states for external memory. These bits are effective only in the 16-bit data bus mode (modes 1, 3, and 5).

Bit 5	Bit 4	Page 0	Page 1	Pages 2 and 3	Pages 4 to 15
0*	0*	3 states	3 states	3 states	3 states
0	1	2 states	3 states	3 states	3 states
1	0	2 states	2 states	3 states	3 states
1	1	2 states	2 states	2 states	3 states

Note: * Initial value.

Bits 3 and 2—Wait Mode Select 1 and 0 (WMS1 and WMS0): These bits select the wait mode.

Bit 3 WMS1	Bit 2 WMS0	Description
0	0	Programmable wait mode (Also, the initial value.)
	1	No wait states are inserted, regardless of the wait count.
1	0	Pin wait mode
	1	Pin auto-wait mode

Bits 1 and 0—Wait Count (WC1 and WC0): These bits specify the number of wait states to be inserted.

Wait states are inserted only in bus cycles in which the CPU or DTC accesses an external address.

Bit 1 WC1	Bit 0 WC0	Description
0	0	No wait states are inserted, except in pin wait mode.
	1	1 wait state in inserted.
1	0	2 wait states are inserted.
	1	3 wait states are inserted. (Also, the initial value.)

7.3 Operation in Each Wait Mode

Table 7.1 Wait Modes

Mode	$\overline{\text{WAIT}}$ Pin Function	Insertion Conditions	Number of Wait States Inserted
Programmable wait mode WMS1 = 0 WMS0 = 0	Disabled	Inserted when accessing an external address	1 to 3 wait states are inserted, as specified by bits WC0 and WC1.
Pin wait mode WMS1 = 1 WMS0 = 0	Enabled	Inserted when accessing an external address	0 to 3 wait states are inserted, as specified by bits WC0 and WC1, plus additional wait states while the $\overline{\text{WAIT}}$ pin is held low.
Pin auto-wait mode WMS1 = 1 WMS0 = 1	Enabled	Inserted when accessing an external address if the $\overline{\text{WAIT}}$ pin is low	1 to 3 wait states are inserted, as specified by bits WC0 and WC1.

7.3.1 Programmable Wait Mode

The programmable wait mode is selected when WMS1 = 0 and WMS0 = 0.

Whenever the CPU, DTC, or ISP accesses an external address, the number of wait states set in bits WC1 and WC0 are inserted (figure 7.2). The $\overline{\text{WAIT}}$ pin is not used for wait control in this mode; it is available as an I/O pin.

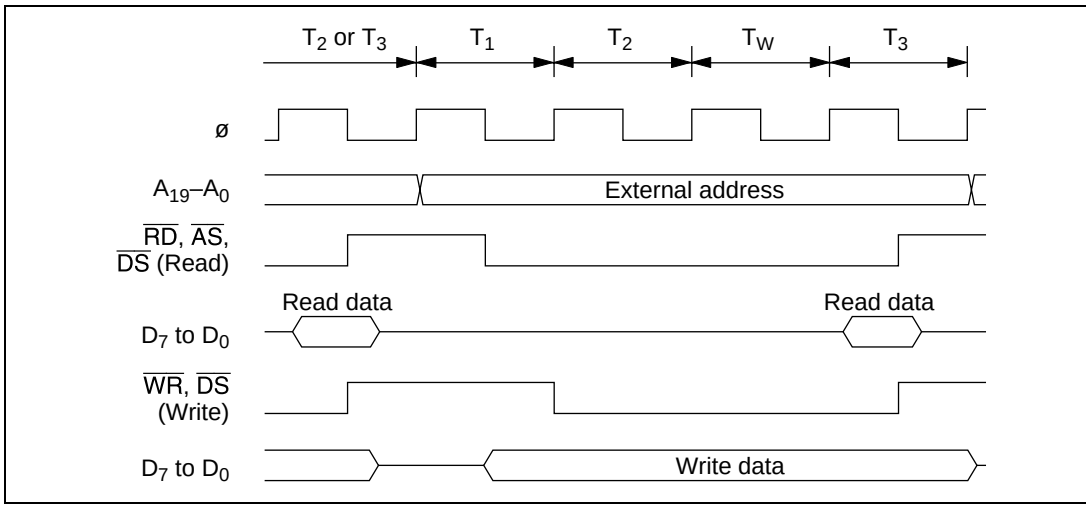


Figure 7.2 Timing of Programmable Wait Mode

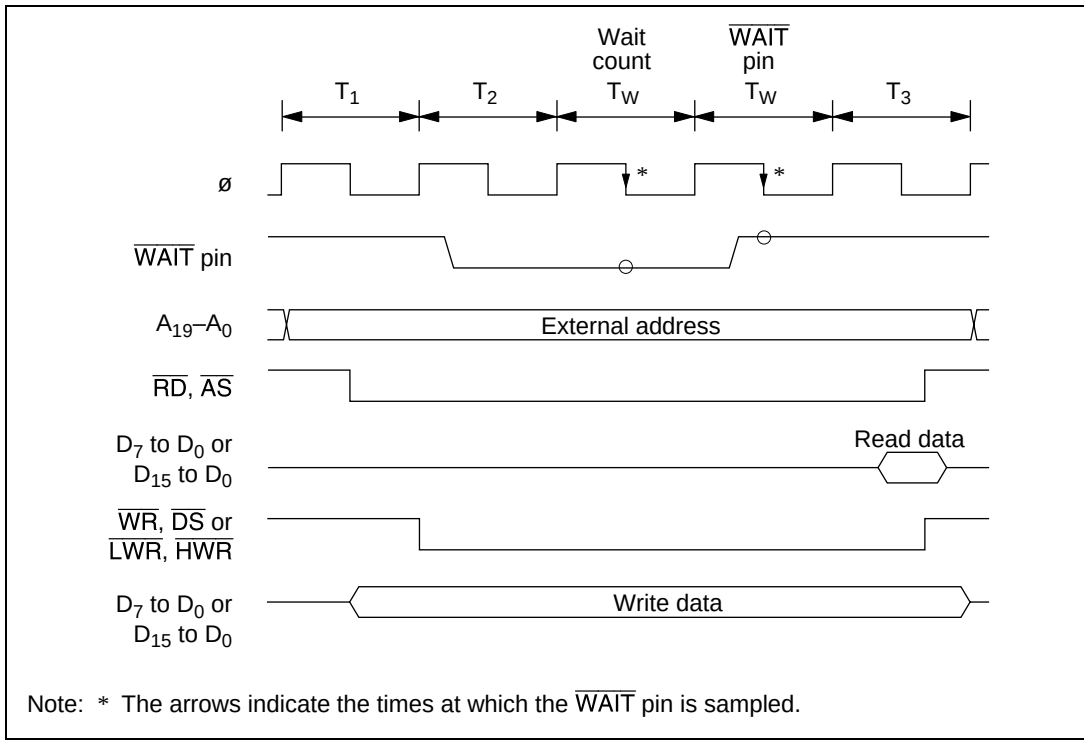
7.3.2 Pin Wait Mode

The pin wait mode is selected when WMS1 = 1 and WMS0 = 0.

In this mode the $\overline{\text{WAIT}}$ pin function is used automatically.

The number of wait states indicated by bits WC1 and WC0 are inserted into any bus cycle in which the CPU, DTC, or ISP accesses an external address. In addition, wait states continue to be inserted as long as the $\overline{\text{WAIT}}$ pin is held low (figure 7.3). In particular, if the wait count is 0 but the $\overline{\text{WAIT}}$ pin is low at the rising edge of the ø clock in the T_2 state, wait states are inserted until the $\overline{\text{WAIT}}$ pin is set high.

This mode is useful for inserting four or more wait states, or when different external devices require different numbers of wait states.



Note: * The arrows indicate the times at which the $\overline{\text{WAIT}}$ pin is sampled.

Figure 7.3 Timing of Pin Wait Mode

7.3.3 Pin Auto-Wait Mode

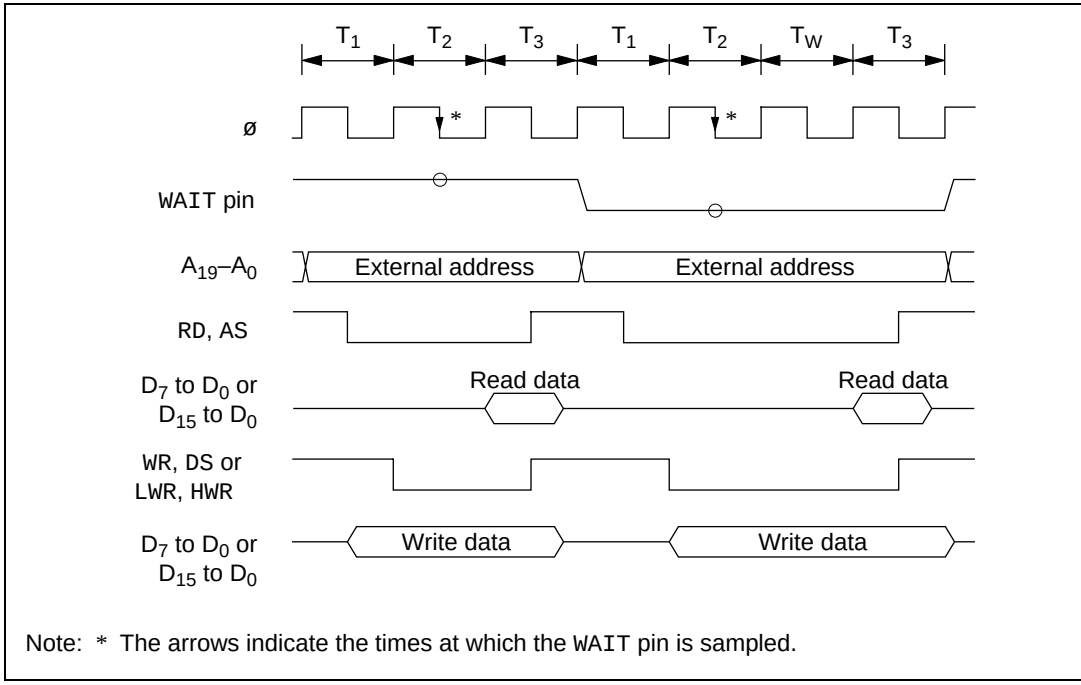
The pin auto-wait mode is selected when WMS1 = 1 and WMS0 = 1.

In this mode the $\overline{\text{WAIT}}$ pin function is used automatically.

In this mode, the number of wait states indicated by bits WC1 and WC0 are inserted only if the $\overline{\text{WAIT}}$ pin is held low (figure 7.4).

In the pin auto-wait mode, the $\overline{\text{WAIT}}$ pin is sampled only once, at the falling edge of the ø clock in the T_2 state. If the $\overline{\text{WAIT}}$ pin is low at this time, the wait-state controller inserts the number of wait states indicated by bits WC1 and WC0. The $\overline{\text{WAIT}}$ pin is not sampled during the T_W and T_3 states, so no additional wait states are inserted even if the $\overline{\text{WAIT}}$ pin continues to be held low.

This mode offers a simple way to interface a low-speed device: the specific number of wait states can be inserted by routing an address decode signal to the $\overline{\text{WAIT}}$ pin.



Note: * The arrows indicate the times at which the WAIT pin is sampled.

Figure 7.4 Timing of Pin Auto-Wait Mode

Section 8 Clock Pulse Generator

8.1 Overview

The H8/570 MCU has a built-in clock pulse generator (CPG) consisting of an oscillator circuit, a system (ø) clock divider, an E-clock divider, and a group of prescalers. The prescalers generate clock signals for the on-chip supporting modules.

8.1.1 Block Diagram

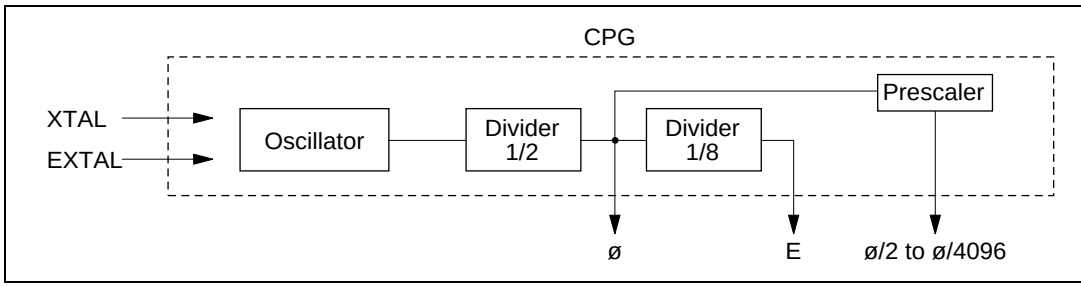


Figure 8.1 Block Diagram of Clock Pulse Generator

8.2 Oscillator Circuit

If an external crystal is connected across the EXTAL and XTAL pins, the on-chip oscillator circuit generates a clock signal for the system clock divider. Alternatively, an external clock signal can be applied to the EXTAL pin.

Connecting an External Crystal

Circuit Configuration: An external crystal can be connected as in the example in figure 8.2. An AT-cut parallel resonating crystal is recommended. The resonance of the crystal should be basic. Do not use an overtone resonant crystal.

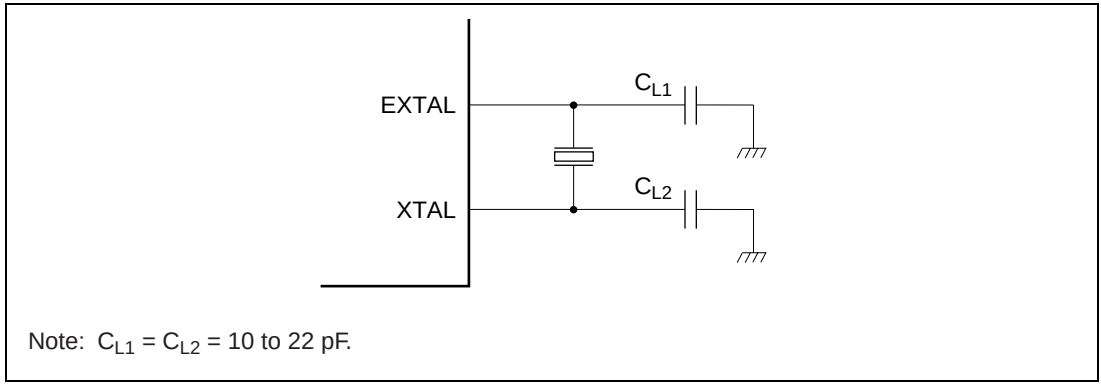


Figure 8.2 Connection of Crystal Oscillator (Example)

Crystal Oscillator: The external crystal must possess the characteristics listed in table 8.1.

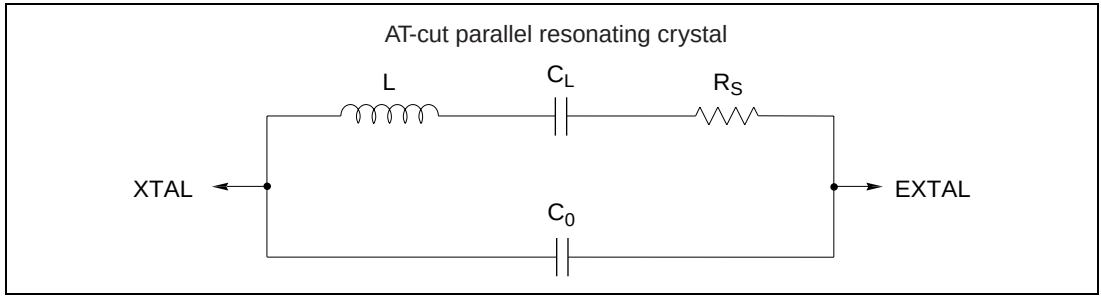


Figure 8.3 Crystal Oscillator Equivalent Circuit

Table 8.1 External Crystal Parameters

Frequency (MHz)	2	4	8	12	16	20	24
R_s max. (Ω)	500	120	60	40	30	20	20
C_0 (pF)	7 pF max.						

Note on Board Design: When an external crystal is connected, other signal lines should be kept away from the crystal circuit to prevent induction from interfering with the correct oscillation. See figure 8.4.

When designing the board, the crystal and its load capacitors should be placed as close as possible to the XTAL and EXTAL pins.

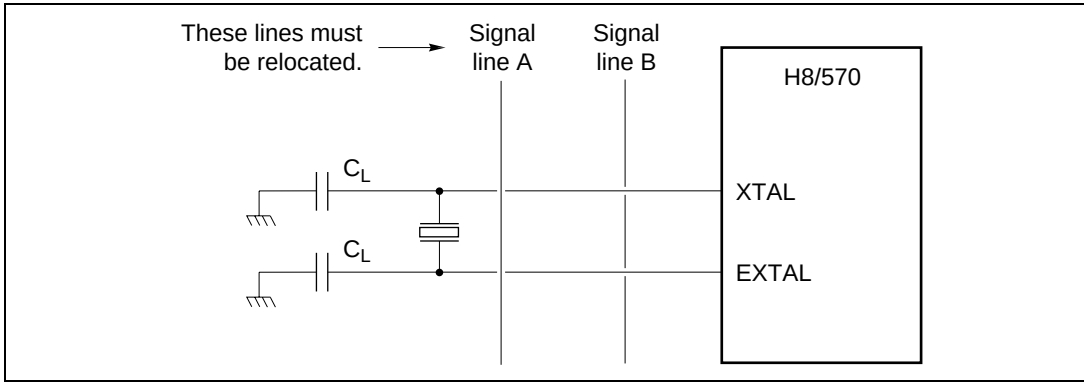


Figure 8.4 Notes on Board Design Around External Crystal

Input of External Clock Signal

Circuit Configuration: An external clock signal can be input at the EXTAL pin (figure 8.5).

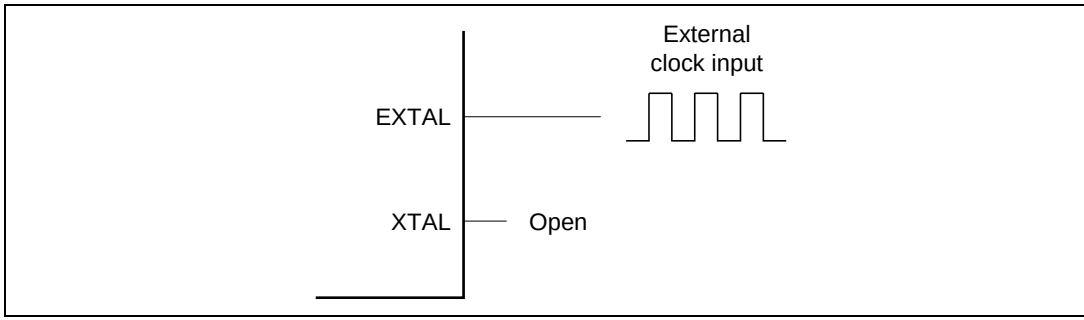


Figure 8.5 External Clock Input (Example)

External Clock Input:

Frequency	Double of the system clock (ø) frequency
Duty factor	45% to 55%

Note on Connection: Leave the XTAL pin open.

8.3 System Clock Divider

The system clock divider divides the crystal oscillator or external clock frequency (f_{OSC}) by 2 to create the ø clock.

An E-clock signal is created by dividing the ø clock by 8 (figure 8.6). The E clock is used for interfacing to E-clock based devices.

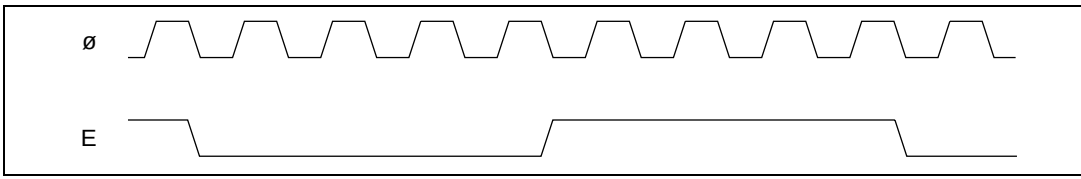


Figure 8.6 Phase Relationship of ø Clock and E Clock

Section 9 I/O Ports

9.1 Overview

The H8/570 has nine ports. Ports 8, 9, 10, 11, and 12 are 8-bit input/output ports. Port 1 is a 5-bit input/output port, port 5 is a 7-bit input/output port, and port 6 is a 6-bit input/output port. Port 7 is an 8-bit input-only port. See table 9.1.

The CPU views each port as a data register (DR) located in the register field at the high addresses of page 0. Each port (except port 7) also has a data direction register (DDR) which determines which pins are used for input or output. Ports 1, 8, 9, and 10 have a system control register n (SYSCRn) for enabling and disabling IRQ_0 and setting other controls.

To read data from an I/O port, the CPU selects input in the data direction register and reads the data register. This causes the input data at the pin to be placed directly on the internal data bus without the use of an intervening input latch.

To send data to an output port, the CPU selects output in the data direction register and writes the desired data in the data register, causing the data to be held in a latch. The latch output then drives the port pins through a buffer amplifier. If the CPU reads the data register of an output port, it obtains the data held in the latch rather than the actual data at the pins.

Outputs from $P1_7$ and $P1_1$ of port 1, $P5_3$–$P5_0$ of port 5, and port 12 can drive one TTL load and a 90-pF capacitive load. Outputs from $P1_3$ and $P1_2$ of port 1, $P5_7$ and $P5_4$ of port 5, port 6, and ports 8 to 11 can drive one TTL load and a 30-pF capacitive load.

Pins $P5_3$ to $P5_0$ have built-in MOS pull-ups that can be controlled by software for each input.

Table 9.1 Input/Output Port Summary

Port	Description	Pins	Mode 1	Mode 3	Mode 4	Mode 5	Mode 6
Port 1	5-bit input/output port	$P1_7$/$\overline{LWR}$	$\overline{LWR}$	$\overline{LWR}$	I/O port	$\overline{LWR}$	I/O port
		$P1_3$/$\overline{WAIT}$ $P1_2$/$\overline{BREQ}$ $P1_1$/$\overline{BACK}$	These pins function as $\overline{WAIT}$, $\overline{BREQ}$, and $\overline{BACK}$ when the necessary control register bits are set to 1. If those bits are set to 0, these pins become I/O ports.				
		$P1_0$/ø	This pin functions as an input pin or clock (ø) output pin depending on the data direction register setting.				
Port 5	7-bit input/output port; Built-in input pull-up (MOS) ($P5_3$ to $P5_0$)	$P5_7$/$\overline{ADTRG}$	$\overline{ADTRG}$ input pin, and input port.				
		$P5_6$/E	Input port if DDR is cleared to 0; clock (E) output pin if DDR is set to 1.				
		$P5_4$/$\overline{IRQ}_0$	$\overline{IRQ}_0$ pin and I/O port.				
		$P5_3$–$P5_0$/ A_{19}–A_{16}	I/O port	Page address bus (A_{19}–A_{16})	I/O port	Page address bus (A_{19}–A_{16})	Page address bus (A_{19}–A_{16})
Port 6	6-bit input/output port	$P6_5$/SCK $P6_4$/RXD $P6_3$/TXD $P6_2$/PW_2 $P6_1$/PW_1 $P6_0$/PW_0	Serial communication interface (SCI) input/output (SCK, RXD, TXD), PWM timer output (PW_2, PW_1, PW_0), and 6-bit I/O port.				
Port 7	8-bit input port	$P7_7$–$P7_0$/ AN_7–AN_0	Analog input pins for A/D converter, and 8-bit input port.				
Port 8	8-bit input/output port	$P8_7$–$P8_0$/ $IOF0_7$–$IOF0_0$	I/O pins for ISP, and 8-bit I/O port.				
Port 9	8-bit input/output port	$P9_7$–$P9_3$/ $IOF1_7$–$IOF1_3$	I/O pins for ISP, and I/O port.				
		$P9_2$–$P9_0$/ PW_5–PW_3/ $IOF1_2$–$IOF1_0$	I/O pins for ISP ($IOF1_2$–$IOF1_0$), PWM timer output pins (PW_5–PW_3), and I/O port.				
Port 10	8-bit input/output port	$P10_7$–$P10_0$/ $IOF2_7$–$IOF2_0$	I/O pins for ISP, and 8-bit I/O port.				
Port 11	8-bit input/output port	$P11_7$–$P11_0$	8-bit I/O port.				
Port 12	8-bit input/output port	$P12_7$–$P12_0$/ D_{15}–D_8	Data bus (D_{15}–D_8)	Data bus (D_{15}–D_8)	I/O port	Data bus (D_{15}–D_8)	I/O port

9.2 Port 1

9.2.1 Overview

Port 1 is a 5-bit input/output port (figure 9.1).

Outputs from $P1_1$ and $P1_3$ of port 1 can drive one TTL load and a 90-pF capacitive load. Also, outputs from $P1_2$ and $P1_3$ can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington transistor pair.

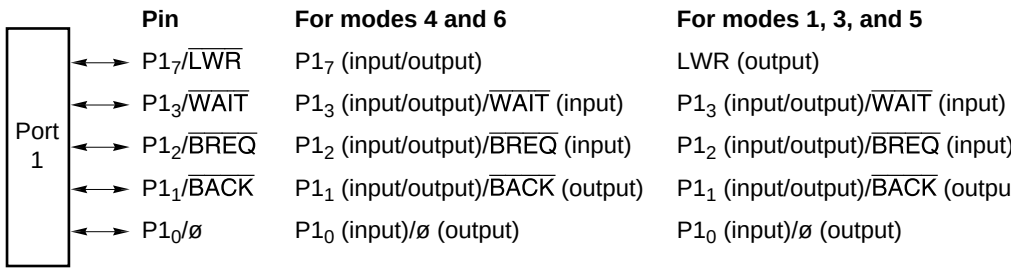


Figure 9.1 Pin Functions of Port 1

9.2.2 Port 1 Registers

Table 9.2 Configuration of Port 1 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 1 data direction register	P1DDR	W	H'71	H'FF2C
Port 1 data register	P1DR	R/W*[1]	Undetermined*[2]	H'FE8C
System control register 1	SYSCR1	R/W	H'87	H'FF4C

Notes: 1. Bit 0 is read-only.
2. Bit 0 is undetermined. Other bits are initialized to 0.

Port 1 Data Direction Register (P1DDR)—H'FF2C

Bit	7	6	5	4	3	2	1	0
	$P1_7DDR$	—	—	—	$P1_3DDR$	$P1_2DDR$	$P1_1DDR$	$P1_0DDR$
Initial value	0	1	1	1	0	0	0	1
Read/Write	W	—	—	—	W	W	W	W

P1DDR is a 5-bit register that selects the direction of each pin in port 1. A pin functions as an output pin if the corresponding bit in P1DDR is set to 1, and as an input pin if the bit is cleared to 0.

P1DDR can only be written to. An attempt to read this register does not cause an error, but all bits will be read as 1, regardless of their true values.

Reset initializes P1DDR to H'71. In the hardware standby mode, P1DDR is cleared to H'00, stopping the clock outputs. P1DDR is not initialized in the software standby mode, so if a P1DDR bit is set to 1 when the MCU enters the software standby mode, the corresponding pin continues to output the value in the port 1 data register.

Port 1 Data Register (P1DR)—H'FE8C

Bit	7	6	5	4	3	2	1	0
	$P1_7$	—	—	—	$P1_3$	$P1_2$	$P1_1$	$P1_0$
Initial value	0	0	0	0	0	0	0	—
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

P1DR is a 5-bit register containing the data for pins $P1_7$ and $P1_3$ to $P1_0$. When the CPU reads P1DR, it reads the value in the P1DR latch for output pins, but for input pins, it obtains the pin status directly.

Note that when pin $P1_0$ is used for output, the clock signal (ø) is output, not the contents of P1DR. If the CPU reads $P1_0$ (when $P1_0DDR = 1$), it obtains the clock values at that instant.

System Control Register 1 (SYSCR1)—H'FF4C

Bit	7	6	5	4	3	2	1	0
	—	—	IRQ_0E	NMIEG	BRLE	—	—	—
Initial value	1	0	0	0	0	1	1	1
Read/Write	—	R/W	R/W	R/W	R/W	—	—	—

SYSCR1 selects the functions of port 1 for bus request and port 5 for external interrupt. It also selects the input edge of the NMI pin.

At reset and in the hardware standby mode, SYSCR1 is initialized to H'87. It is not initialized in the software standby mode.

Bit 7—Reserved: This bit cannot be modified and is always read as 1.

Bit 6—Reserved: The CPU can read or write to this bit.

Bit 5—Interrupt Request 0 Enable (IRQ_0E): This bit selects the function of pin $P5_4$.

Bit 5 IRQ_0E	Description
0	$P5_4$ functions as an I/O pin. (Also, the initial value.)
1	$P5_4$ functions as the $\overline{IRQ}_0$ input pin, regardless of the value set in $P5_4$DDR. (However, the CPU can still read the pin status by reading P5DR.)

Bit 4—Nonmaskable Interrupt Edge (NMIEG): This bit selects the input edge of the NMI pin.

Bit 4 NMIEG	Description
0	A nonmaskable interrupt is generated on the falling edge of the input at the NMI pin. (Also, the initial value.)
1	A nonmaskable interrupt is generated on the rising edge of the input at the NMI pin.

Bit 3—Bus Release Enable (BRLE): This bit selects the functions of pins $P1_2$ and $P1_1$.

Bit 3 BRLE	Description
0	$P1_2$ and $P1_1$ function as I/O pins. (Also, the initial value.)
1	$P1_2$ functions as the $\overline{BREQ}$ input pin. $P1_1$ functions as the $\overline{BACK}$ output pin.

Bits 2 to 0—Reserved: These bits cannot be modified and are always read as 1.

9.2.3 Pin Functions in Each Mode

Port 1 operates differently in modes 4 and 6, and modes 1, 3, and 5. See tables 9.3 and 9.4.

Table 9.3 Port 1 Pin Functions in Modes 4 and 6

Pin	Pin Functions and Function Select Configuration
$P1_7$	$P1_7$DDR: 0 → $P1_7$ input; 1 → $P1_7$ output
$P1_3$/$\overline{WAIT}$	The function depends on the wait mode select 1 bit (WMS1) of the wait control register (WCR) and the $P1_3$DDR bit as follows: WMS1 = 0: $P1_3$DDR 0 → $P1_3$ input; 1 → $P1_3$ output WMS1 = 1: $P1_3$DDR 0 or 1 → $\overline{WAIT}$ input
$P1_2$/$\overline{BREQ}$	The function depends on the BRLE bit and the $P1_2$DDR bit as follows: BRLE = 0: $P1_2$DDR 0 → $P1_2$ input; 1 → $P1_2$ output BRLE = 1: $P1_2$DDR 0 or 1 → $\overline{BREQ}$ input
$P1_1$/$\overline{BACK}$	The function depends on the BRLE bit and the $P1_1$DDR bit as follows: BRLE = 0: $P1_1$DDR 0 → $P1_1$ input; 1 → $P1_1$ output BRLE = 1: $P1_1$DDR 0 or 1 → $\overline{BACK}$ output
$P1_0$/ø	$P1_0$DDR: 0 → $P1_0$ input; 1 → ø clock output

Table 9.4 Port 1 Pin Functions in Modes 1, 3, and 5

Pin	Pin Functions and Function Select Configuration
$P1_7$	$P1_7$DDR is automatically set as output, and this pin functions as $\overline{LWR}$ output.
$P1_3/\overline{WAIT}$	The function depends on the wait mode select 1 bit (WMS1) of the wait control register (WCR) and the $P1_3$DDR bit as follows:

WMS1	0		1	
$P1_3$DDR	0	1	0	1
Pin function	$P1_3$ input	$P1_3$ output	$\overline{WAIT}$ input	

$P1_2/\overline{BREQ}$	The function depends on the BRLE bit and the $P1_2$DDR bit as follows:

BRLE	0		1	
$P1_2$DDR	0	1	0	1
Pin function	$P1_2$ input	$P1_2$ output	$\overline{BREQ}$ input	

$P1_1/\overline{BACK}$	The function depends on the BRLE bit and the $P1_1$DDR bit as follows:

BRLE	0		1	
$P1_1$DDR	0	1	0	1
Pin function	$P1_1$ input	$P1_1$ output	$\overline{BACK}$ output	

$P1_0$/ø

$P1_0$DDR	0	1
Pin function	$P1_0$ input	ø clock output

9.3 Port 5

9.3.1 Overview

Port 5 is a 7-bit input/output port (figure 9.2).

Pins $P5_3$ to $P5_0$ have built-in MOS pull-ups that can be turned on or off by the program.

Outputs $P5_3$ to $P5_0$ of port 5 can drive one TTL load and a 90-pF capacitive load. Also, outputs $P5_7$ and $P5_4$ can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington transistor pair.

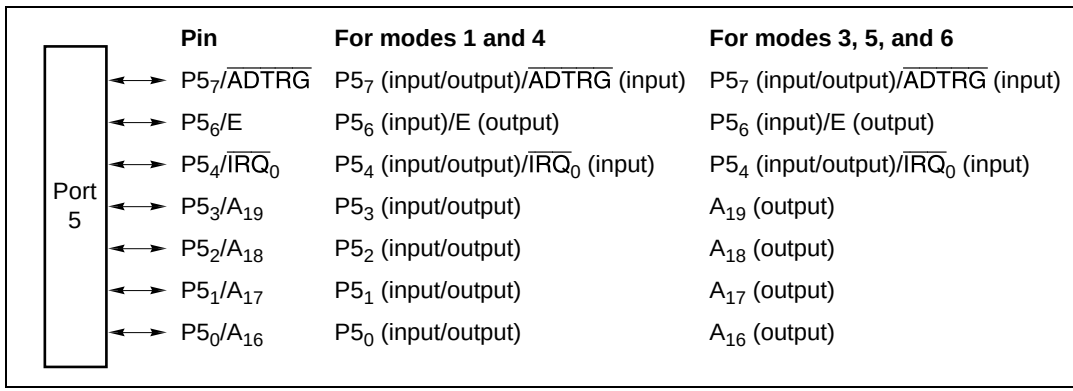


Figure 9.2 Pin Functions of Port 5

9.3.2 Port 5 Registers

Table 9.5 Configuration of Port 5 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 5 data direction register	P5DDR	W	H'60	H'FF30
Port 5 data register	P5DR	R/W*[1]	Undetermined*[2]	H'FE90

Notes: 1. Bit 6 is read-only.
2. Bits 5 and 6 are undetermined. Other bits are initialized to 0.

Port 5 Data Direction Register (P5DDR)—H'FF30

Bit	7	6	5	4	3	2	1	0
	$P5_7DDR$	$P5_6DDR$	—	$P5_4DDR$	$P5_3DDR$	$P5_2DDR$	$P5_1DDR$	$P5_0DDR$
Initial value	0	1	1	0	0	0	0	0
Read/Write	W	W	—	W	W	W	W	W

P5DDR is a 7-bit register that selects the direction of each pin in port 5. A pin functions as an output pin if the corresponding bit in P5DDR is set to 1, and as an input pin if the bit is cleared to 0.

P5DDR can only be written to. An attempt to read this register does not cause an error, but all bits will be read as 1, regardless of their true values. In mode 5, pins $P5_3$ to $P5_0$ output addresses, and the corresponding bits of $P5_3$DDR to $P5_0$DDR are fixed at 1.

Reset initializes P5DDR to H'60. In the hardware standby mode, P5DDR is initialized to H'20. P5DDR is not initialized in the software standby mode, so if a P5DDR bit is set to 1 when the MCU enters the software standby mode, the corresponding pin continues to output the value in the port 5 data register.

If pin $P5_7$ is used as $\overline{\text{ADTRG}}$ input when the MCU enters the software standby mode, $P5_7$ changes to an input/output port depending on values $P5_7$DDR and $P5_7$DR since the A/D converter is initialized.

Port 5 Data Register (P5DR)—H'FE90

Bit	7	6	5	4	3	2	1	0
	$P5_7$	$P5_6$	—	$P5_4$	$P5_3$	$P5_2$	$P5_1$	$P5_0$
Initial value	0	—	1	0	0	0	0	0
Read/Write	R/W	R	—	R/W	R/W	R/W	R/W	R/W

P5DR is a 7-bit register containing the data for pins $P5_7$, $P5_6$, and $P5_4$ to $P5_0$.

When the CPU reads P5DR, it reads the value in the P5DR latch for output pins, but for input pins, it obtains the pin status directly.

Note that when pin $P5_6$ is used for output, the E-clock signal (E) is output, not the contents of P5DR. If the CPU reads $P5_6$ (when $P5_6$CCR = 1), it obtains the clock values at that instant.

9.3.3 Pin Functions in Each Mode

Port 5 operates differently in modes 1 and 4, and modes 3, 5, and 6. See tables 9.6 and 9.7.

Table 9.6 Port 5 Pin Functions in Modes 1 and 4

<table>
<tr><th>Pin</th><th colspan="4">Pin Functions and Function Select Configuration</th></tr>
<tr><td>$P5_7/\overline{ADTRG}$</td><td colspan="4">The function depends on the trigger enable bit (TRGE) of the A/D control register (ADCR) in the A/D converter and the $P5_7$DDR bit as follows:
<table><tr><td>TRGE</td><td colspan="2">0</td><td colspan="2">1</td></tr><tr><td>$P5_7$DDR</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>Pin function</td><td>$P5_7$ input</td><td>$P5_7$ output</td><td colspan="2">$\overline{ADTRG}$ input/$P5_7$ input</td></tr></table></td></tr>
<tr><td>$P5_6$/E</td><td colspan="4"><table><tr><td>$P5_6$DDR</td><td>0</td><td>1</td></tr><tr><td>Pin function</td><td>$P5_6$ input</td><td>E-clock output</td></tr></table></td></tr>
<tr><td>$P5_4/\overline{IRQ_0}$</td><td colspan="4">The function depends on the IRQ_0E bit and the $P5_4$DDR bit as follows:
<table><tr><td>IRQ_0E</td><td colspan="2">0</td><td colspan="2">1</td></tr><tr><td>$P5_4$DDR</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>Pin function</td><td>$P5_4$ input</td><td>$P5_4$ output</td><td colspan="2">$\overline{IRQ_0}$ input</td></tr></table></td></tr>
<tr><td>$P5_3$</td><td colspan="4"><table><tr><td>$P5_3$DDR</td><td>0</td><td>1</td></tr><tr><td>Pin function</td><td>$P5_3$ input</td><td>$P5_3$ output</td></tr></table></td></tr>
<tr><td>$P5_2$</td><td colspan="4"><table><tr><td>$P5_2$DDR</td><td>0</td><td>1</td></tr><tr><td>Pin function</td><td>$P5_2$ input</td><td>$P5_2$ output</td></tr></table></td></tr>
<tr><td>$P5_1$</td><td colspan="4"><table><tr><td>$P5_1$DDR</td><td>0</td><td>1</td></tr><tr><td>Pin function</td><td>$P5_1$ input</td><td>$P5_1$ output</td></tr></table></td></tr>
<tr><td>$P5_0$</td><td colspan="4"><table><tr><td>$P5_0$DDR</td><td>0</td><td>1</td></tr><tr><td>Pin function</td><td>$P5_0$ input</td><td>$P5_0$ output</td></tr></table></td></tr>
</table>

Table 9.7 Port 5 Pin Functions in Modes 3, 5, and 6

Pin	Pin Functions and Function Select Configuration
$P5_7/\overline{ADTRG}$	The function depends on the trigger enable bit (TRGE) of the A/D control register (ADCR) in the A/D converter and the $P5_7$DDR bit as follows:

TRGE	0		1	
$P5_7$DDR	0	1	0	1
Pin function	$P5_7$ input	$P5_7$ output	$\overline{ADTRG}$ input/$P5_7$ input	

$P5_6$/E	$P5_6$DDR	0	1
	Pin function	$P5_6$ input	E-clock output

$P5_4/\overline{IRQ_0}$ The function depends on the IRQ_0E bit and the $P5_4$DDR bit as follows:

IRQ_0E	0		1	
$P5_4$DDR	0	1	0	1
Pin function	$P5_4$ input	$P5_4$ output	$\overline{IRQ_0}$ input	

A_{19}	$P5_3$DDR is automatically set as output, and this pin functions as A_{19} output.
A_{18}	$P5_2$DDR is automatically set as output, and this pin functions as A_{18} output.
A_{17}	$P5_1$DDR is automatically set as output, and this pin functions as A_{17} output.
A_{16}	$P5_0$DDR is automatically set as output, and this pin functions as A_{16} output.

9.3.4 Built-In MOS Pull-Up

The MOS input pull-up of pins $P5_3$ to $P5_0$ is turned on by clearing the corresponding bit in P5DDR to 0 and writing a 1 in P5DR. This pull-up is turned off at reset and in the hardware standby mode (table 9.8).

Table 9.8 Status of MOS Pull-Up for Port 5

Reset	Hardware Standby Mode	Other Operating States*
Off	Off	On/Off
Off	Off	Off

Notes: * Including the software standby mode.
Off: The MOS pull-up is always off.
On/Off: The MOS pull-up is on when P5DDR = 0 and P5DR = 1, and off otherwise.

9.4 Port 6

9.4.1 Overview

Port 6 is a 6-bit input/output port (figure 9.3). Its pins also input and output signals for the serial communication interface (SCI), and output signals for the PWM timer. The pin functions are the same in all MCU operating modes.

Outputs from port 6 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington transistor pair.

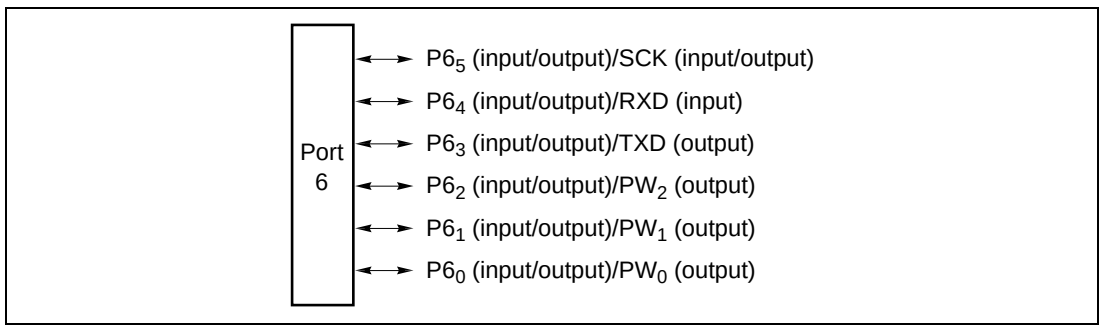


Figure 9.3 Pin Functions of Port 6

9.4.2 Port 6 Registers

Table 9.9 Configuration of Port 6 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 6 data direction register	P6DDR	W	H'C0	H'FF31
Port 6 data register	P6DR	R/W	H'C0	H'FE91

Port 6 Data Direction Register (P6DDR)—H'FF31

Bit	7	6	5	4	3	2	1	0
	—	—	$P6_5$DDR	$P6_4$DDR	$P6_3$DDR	$P6_2$DDR	$P6_1$DDR	$P6_0$DDR
Initial value	1	1	0	0	0	0	0	0
Read/Write	—	—	W	W	W	W	W	W

P6DDR is a 6-bit register that selects the direction of each pin in port 6. A pin functions as an output pin if the corresponding bit in P6DDR is set to 1, and as an input pin if the bit is cleared to 0.

Bits 5 to 0 can only be written to. An attempt to read these bits does not cause an error, but all bits will be read as 1, regardless of their true values.

Bits 7 and 6 are reserved. They cannot be modified and are always read as 1.

At reset and in the hardware standby mode, P6DDR is initialized to H'C0. P6DDR is not initialized in the software standby mode, so if a P6DDR bit is set to 1 when the MCU enters the software standby mode, the corresponding pin continues to output the value in the port 6 data register.

If port 6 is used as pins for an on-chip supporting module when the MCU enters the software standby mode, port 6 changes to an input/output port depending on values P6DDR and P6DR since the on-chip supporting module is initialized.

Port 6 Data Register (P6DR)—H'FE91

Bit	7	6	5	4	3	2	1	0
	—	—	$P6_5$	$P6_4$	$P6_3$	$P6_2$	$P6_1$	$P6_0$
Initial value	1	1	0	0	0	0	0	0
Read/Write	—	—	R/W	R/W	R/W	R/W	R/W	R/W

P6DR is a 6-bit register containing the data for pins $P6_5$ to $P6_0$.

Bits 7 and 6 are reserved. They cannot be modified and are always read as 1.

When the CPU reads P6DR, it reads the value in the P6DR latch for output pins, but for input pins, it obtains the pin status directly.

9.4.3 Pin Functions in Each Mode

All pins of port 6 have dual functions (table 9.10).

Table 9.10 Port 6 Pin Functions

Pin	Pin Functions and Function Select Configuration
$P6_5$/SCK	The function depends on the communication mode bit (C/$\overline{A}$) of the serial mode register (SMR) and the clock enable bits 1 and 0 (CKE1 and CKE0) of the serial control register (SCR) for the serial communication interface (SCI) as follows:

C/$\overline{A}$	0				1			
CKE1	0		1		0		1	
CKE0	0	1	0	1	0	1	0	1
Pin function	$P6_5$ input/ output	SCI internal clock output	SCI external clock input		SCI internal clock output		SCI external clock input	

When this pin functions as $P6_5$ input/output, $P6_5$DDR determines if this pin is used for input or output.

$P6_4$/RXD — The function depends on the receive enable bit (RE) of the serial control register (SCR) for the SCI and the $P6_4$DDR bit as follows:

RE	0		1	
$P6_4$DDR	0	1	0	1
Pin function	$P6_4$ input	$P6_4$ output	RXD input	

$P6_3$/TXD — The function depends on the transmit enable bit (TE) of the serial control register (SCR) for the SCI and the $P6_3$DDR bit as follows:

TE	0		1	
$P6_3$DDR	0	1	0	1
Pin function	$P6_3$ input	$P6_3$ output	TXD output	

$P6_2$/PW_2 — The function depends on the PW_2 output specification by the timer control register (TCR) of the PWM timer and the $P6_2$DDR bit as follows:

PW_2 output specification	Output disable		Output enable	
$P6_2$DDR	0	1	0	1
Pin function	$P6_2$ input	$P6_2$ output	PW_2 output	

Pin	Pin Functions and Function Select Configuration
$P6_1/PW_1$	The function depends on the PW_1 output specification by the timer control register (TCR) of the PWM timer and the $P6_1$DDR bit as follows:

PW_1 output specification	Output disable		Output enable	
$P6_1$DDR	0	1	0	1
Pin function	$P6_1$ input	$P6_1$ output	PW_1 output	

$P6_0/PW_0$	The function depends on the PW_0 output specification by the timer control register (TCR) of the PWM timer and the $P6_0$DDR bit as follows:

PW_0 output specification	Output disable		Output enable	
$P6_0$DDR	0	1	0	1
Pin function	$P6_0$ input	$P6_0$ output	PW_0 output	

9.5 Port 7

9.5.1 Overview

Port 7 is an 8-bit input port (figure 9.4). Its pins also input signals for the A/D converter. The pin functions are the same in all MCU operating modes.

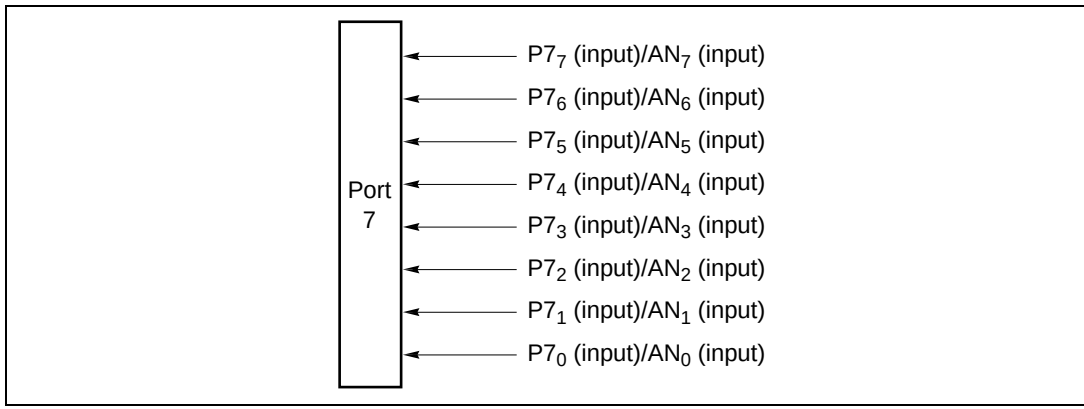


Figure 9.4 Pin Functions of Port 7

9.5.2 Port 7 Register

Table 9.11 Port 7 Register Configuration

Name	Abbreviation	Read/Write	Address
Port 7 data register	P7DR	R	H'FE92

Port 7 Data Register (P7DR)—H'FE92

Bit	7	6	5	4	3	2	1	0
	P7$_7$	P7$_6$	P7$_5$	P7$_4$	P7$_3$	P7$_2$	P7$_1$	P7$_0$
Read/Write	R	R	R	R	R	R	R	R

P7DR is an 8-bit register containing the data for pins P7$_7$ to P7$_0$. When the CPU reads P7DR, it obtains the pin status directly.

9.6 Port 8

9.6.1 Overview

Port 8 is an 8-bit input/output port (figure 9.5). Its pins also input and output signals for the Intelligent Sub-Processor (ISP). The pin functions are the same in all MCU operating modes.

Outputs from port 8 can drive one TTL load and a 30-pF capacitive load.

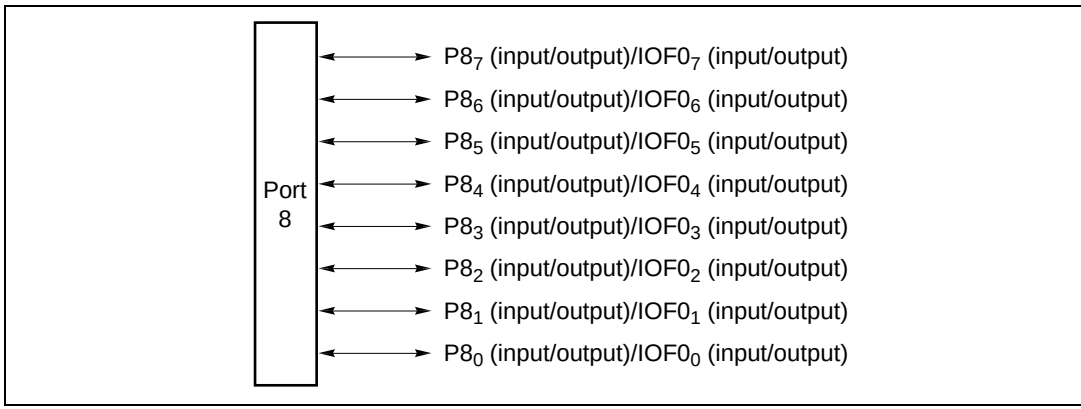


Figure 9.5 Pin Functions of Port 8

9.6.2 Port 8 Registers

Table 9.12 Configuration of Port 8 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 8 data direction register	P8DDR	W	H'00	H'FF33
Port 8 data register	P8DR	R/W	H'00	H'FE93
System control register 8	SYSCR8	R/W	H'00	H'FF23

Port 8 Data Direction Register (P8DDR)—H'FF33

Bit	7	6	5	4	3	2	1	0
	$P8_7DDR$	$P8_6DDR$	$P8_5DDR$	$P8_4DDR$	$P8_3DDR$	$P8_2DDR$	$P8_1DDR$	$P8_0DDR$
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

P8DDR is an 8-bit register that selects the direction of each pin in port 8. A pin functions as an output pin if the corresponding bit in P8DDR is set to 1, and as an input pin if the bit is cleared to 0.

P8DDR can only be written to. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

At reset and in the hardware standby mode, P8DDR is cleared to H'00, stopping the clock outputs. P8DDR is not initialized in the software standby mode, so if a P8DDR bit is set to 1 when the MCU enters the software standby mode, the corresponding pin continues to output the value in the port 8 data register.

Port 8 Data Register (P8DR)—H'FE93

Bit	7	6	5	4	3	2	1	0
	$P8_7$	$P8_6$	$P8_5$	$P8_4$	$P8_3$	$P8_2$	$P8_1$	$P8_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P8DR is an 8-bit register containing the data for pins $P8_7$ to $P8_0$. When the CPU reads P8DR, it reads the value in the P8DR latch for output pins, but for input pins, it obtains the pin status directly.

System Control Register 8 (SYSCR8)—H'FF23

Bit	7	6	5	4	3	2	1	0
	$IOF0_7E$	$IOF0_6E$	$IOF0_5E$	$IOF0_4E$	$IOF0_3E$	$IOF0_2E$	$IOF0_1E$	$IOF0_0E$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

SYSCR8 selects the functions of the port 8 pins.

At reset and in the hardware standby mode, SYSCR8 is initialized to H'00. It is not initialized in the software standby mode.

Bit 7—IOF0$_7$ Enable (IOF0$_7$E): This bit selects the function of the P8$_7$ pin.

Bit 7 IOF0$_7$E	Description
0	P8$_7$ functions as an I/O pin. (Also, the initial value.)
1	P8$_7$ functions as an I/O pin of the ISP (IOF0$_7$).

Bit 6—IOF0$_6$ Enable (IOF0$_6$E): This bit selects the function of the P8$_6$ pin.

Bit 6 IOF0$_6$E	Description
0	P8$_6$ functions as an I/O pin. (Also, the initial value.)
1	P8$_6$ functions as an I/O pin of the ISP (IOF0$_6$).

Bit 5—IOF0$_5$ Enable (IOF0$_5$E): This bit selects the function of the P8$_5$ pin.

Bit 5 IOF0$_5$E	Description
0	P8$_5$ functions as an I/O pin. (Also, the initial value.)
1	P8$_5$ functions as an I/O pin of the ISP (IOF0$_5$).

Bit 4—IOF0$_4$ Enable (IOF0$_4$E): This bit selects the function of the P8$_4$ pin.

Bit 4 IOF0$_4$E	Description
0	P8$_4$ functions as an I/O pin. (Also, the initial value.)
1	P8$_4$ functions as an I/O pin of the ISP (IOF0$_4$).

Bit 3—IOF0$_3$ Enable (IOF0$_3$E): This bit selects the function of the P8$_3$ pin.

Bit 3 IOF0$_3$E	Description
0	P8$_3$ functions as an I/O pin. (Also, the initial value.)
1	P8$_3$ functions as an I/O pin of the ISP (IOF0$_3$).

Bit 2—IOF0$_2$ Enable (IOF0$_2$E): This bit selects the function of the P8$_2$ pin.

Bit 2 IOF0$_2$E	Description
0	P8$_2$ functions as an I/O pin. (Also, the initial value.)
1	P8$_2$ functions as an I/O pin of the ISP (IOF0$_2$).

Bit 1—IOF0$_1$ Enable (IOF0$_1$E): This bit selects the function of the P8$_1$ pin.

Bit 1 IOF0$_1$E	Description
0	P8$_1$ functions as an I/O pin. (Also, the initial value.)
1	P8$_1$ functions as an I/O pin of the ISP (IOF0$_1$).

Bit 0—IOF0$_0$ Enable (IOF0$_0$E): This bit selects the function of the P8$_0$ pin.

Bit 0 IOF0$_0$E	Description
0	P8$_0$ functions as an I/O pin. (Also, the initial value.)
1	P8$_0$ functions as an I/O pin of the ISP (IOF0$_0$).

9.7 Port 9

9.7.1 Overview

Port 9 is an 8-bit input/output port (figure 9.6). Its pins also input and output signals for the Intelligent Sub-Processor (ISP), and output signals for the PWM timer. The pin functions are the same in all MCU operating modes.

Outputs from port 9 can drive one TTL load and a 30-pF capacitive load.

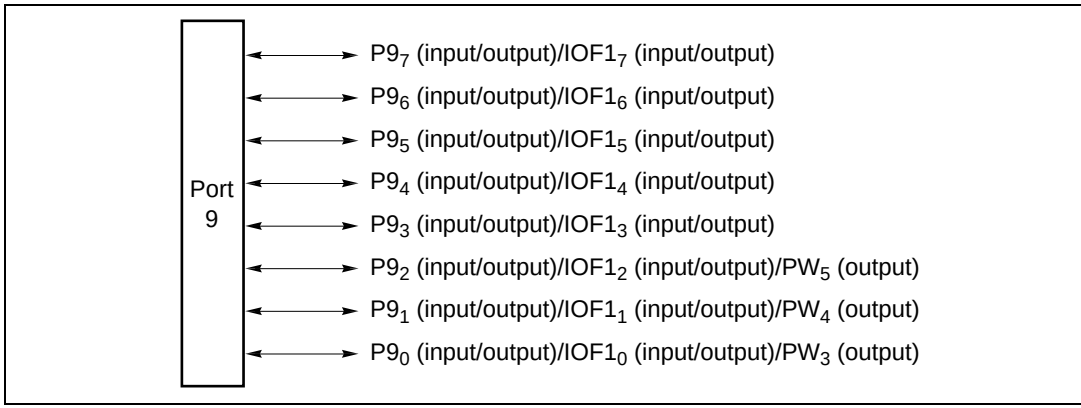


Figure 9.6 Pin Functions of Port 9

9.7.2 Port 9 Registers

Table 9.13 Configuration of Port 9 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 9 data direction register	P9DDR	W	H'00	H'FF34
Port 9 data register	P9DR	R/W	H'00	H'FE94
System control register 9	SYSCR9	R/W	H'00	H'FF24

Port 9 Data Direction Register (P9DDR)—H'FF34

Bit	7	6	5	4	3	2	1	0
	$P9_7$DDR	$P9_6$DDR	$P9_5$DDR	$P9_4$DDR	$P9_3$DDR	$P9_2$DDR	$P9_1$DDR	$P9_0$DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

P9DDR is an 8-bit register that selects the direction of each pin in port 9. A pin functions as an output pin if the corresponding bit in P9DDR is set to 1, and as an input pin if the bit is cleared to 0.

P9DDR can only be written to. An attempt to read this register does not cause an error, but all bits will be read as 1, regardless of their true values.

At reset and in the hardware standby mode, P9DDR is initialized to H'00, setting all pins for input. P9DDR is not initialized in the software standby mode, so if a P9DDR bit is set to 1 when the MCU enters the software standby mode, the corresponding pin continues to output the value in the port 9 data register.

When $P9_2$ to $P9_0$ are specified as PWM output pins by a PWM timer register, pins $P9_2$ to $P9_0$ function as PW_5 to PW_3 outputs regardless of P9DDR values. However, if the MCU enters the software standby mode in this state, pins $P9_2$ to $P9_0$ change to an input/output port depending on values P9DDR, P9DR, and SYSCR9 since the PWM timer module is initialized.

Port 9 Data Register (P9DR)—H'FE94

Bit	7	6	5	4	3	2	1	0
	$P9_7$	$P9_6$	$P9_5$	$P9_4$	$P9_3$	$P9_2$	$P9_1$	$P9_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P9DR is an 8-bit register containing the data for pins $P9_7$ to $P9_0$. When the CPU reads P9DR, it reads the value in the P9DR latch for output pins, but for input pins, it obtains the pin status directly.

System Control Register 9 (SYSCR9)—H'FF24

Bit	7	6	5	4	3	2	1	0
	$IOF1_7E$	$IOF1_6E$	$IOF1_5E$	$IOF1_4E$	$IOF1_3E$	$IOF1_2E$	$IOF1_1E$	$IOF1_0E$
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

SYSCR9 selects the functions of the port 9 pins.

At reset and in the hardware standby mode, SYSCR9 is initialized to H'00. It is not initialized in the software standby mode.

Bit 7—$IOF1_7$ Enable ($IOF1_7E$): This bit selects the function of the $P9_7$ pin.

Bit 7 $IOF1_7E$	Description
0	$P9_7$ functions as an I/O pin. (Also, the initial value.)
1	$P9_7$ functions as an I/O pin of the ISP ($IOF1_7$).

Bit 6—$IOF1_6$ Enable ($IOF1_6E$): This bit selects the function of the $P9_6$ pin.

Bit 6 $IOF1_6E$	Description
0	$P9_6$ functions as an I/O pin. (Also, the initial value.)
1	$P9_6$ functions as an I/O pin of the ISP ($IOF1_6$).

Bit 5—$IOF1_5$ Enable ($IOF1_5E$): This bit selects the function of the $P9_5$ pin.

Bit 5 $IOF1_5E$	Description
0	$P9_5$ functions as an I/O pin. (Also, the initial value.)
1	$P9_5$ functions as an I/O pin of the ISP ($IOF1_5$).

Bit 4—$IOF1_4$ Enable ($IOF1_4E$): This bit selects the function of the $P9_4$ pin.

Bit 4 $IOF1_4E$	Description
0	$P9_4$ functions as an I/O pin. (Also, the initial value.)
1	$P9_4$ functions as an I/O pin of the ISP ($IOF1_4$).

Bit 3—IOF1$_3$ Enable (IOF1$_3$E): This bit selects the function of the P9$_3$ pin.

Bit 3 IOF1$_3$E	Description
0	P9$_3$ functions as an I/O pin. (Also, the initial value.)
1	P9$_3$ functions as an I/O pin of the ISP (IOF1$_3$).

Bit 2—IOF1$_2$ Enable (IOF1$_2$E): This bit selects the function of the P9$_2$ pin. However, if the P9$_2$ pin is specified as PW$_5$ output by the PWM timer register, the value of this bit is ignored.

Bit 2 IOF1$_2$E	Description
0	P9$_2$ functions as an I/O pin. (Also, the initial value.)
1	P9$_2$ functions as an I/O pin of the ISP (IOF1$_2$).

Bit 1—IOF1$_1$ Enable (IOF1$_1$E): This bit selects the function of the P9$_1$ pin. However, if the P9$_1$ pin is specified as PW$_4$ output by the PWM timer register, the value of this bit is ignored.

Bit 1 IOF1$_1$E	Description
0	P9$_1$ functions as an I/O pin. (Also, the initial value.)
1	P9$_1$ functions as an I/O pin of the ISP (IOF1$_1$).

Bit 0—IOF1$_0$ Enable (IOF1$_0$E): This bit selects the function of the P9$_0$ pin. However, if the output P9$_0$ pin is specified as PW$_3$ output by the PWM timer register, the value of this bit is ignored.

Bit 0 IOF1$_0$E	Description
0	P9$_0$ functions as an I/O pin. (Also, the initial value.)
1	P9$_0$ functions as an I/O pin of the ISP (IOF1$_0$).

9.8 Port 10

9.8.1 Overview

Port 10 is an 8-bit input/output port (figure 9.7). Its pins also input and output signals for the Intelligent Sub-Processor (ISP). The pin functions are the same in all MCU operating modes.

Outputs from port 10 can drive one TTL load and a 30-pF capacitive load.

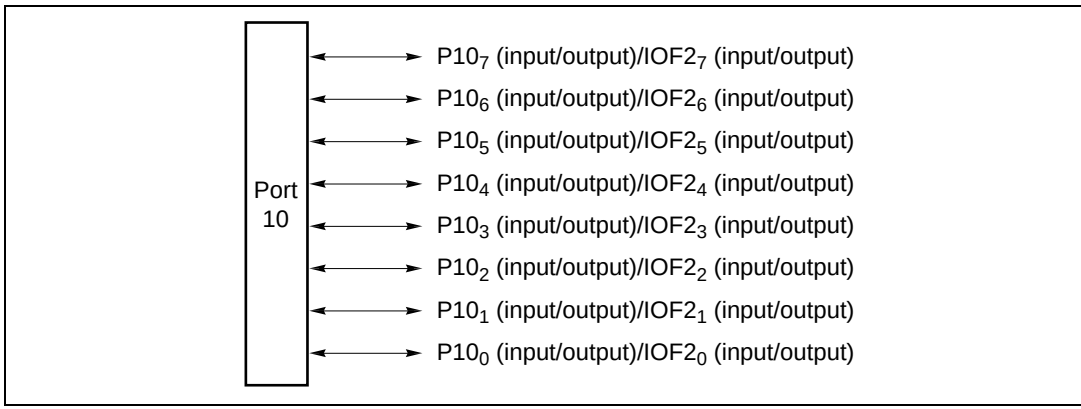


Figure 9.7 Pin Functions of Port 10

9.8.2 Port 10 Registers

Table 9.14 Configuration of Port 10 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 10 data direction register	P10DDR	W	H'00	H'FF35
Port 10 data register	P10DR	R/W	H'00	H'FE95
System control register 10	SYSCR10	R/W	H'00	H'FF25

Port 10 Data Direction Register (P10DDR)—H'FF35

Bit	7	6	5	4	3	2	1	0
	$P10_7$DDR	$P10_6$DDR	$P10_5$DDR	$P10_4$DDR	$P10_3$DDR	$P10_2$DDR	$P10_1$DDR	$P10_0$DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

P10DDR is an 8-bit register that selects the direction of each pin in port 10. A pin functions as an output pin if the corresponding bit in P10DDR is set to 1, and as an input pin if the bit is cleared to 0.

P10DDR can only be written to. An attempt to read this register does not cause an error, but all bits will be read as 1, regardless of their true values.

At reset and in the hardware standby mode, P10DDR is initialized to H'00, setting all pins for input. P10DDR is not initialized in the software standby mode, so if a P10DDR bit is set to 1 when the MCU enters the software standby mode, the corresponding pin continues to output the value in the port 10 data register.

Port 10 Data Register (P10DR)—H'FE95

Bit	7	6	5	4	3	2	1	0
	$P10_7$	$P10_6$	$P10_5$	$P10_4$	$P10_3$	$P10_2$	$P10_1$	$P10_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P10DR is an 8-bit register containing the data for pins $P10_7$ to $P10_0$. When the CPU reads P10DR, it reads the value in the P10DR latch for output pins, but for input pins, it obtains the pin status directly.

System Control Register 10 (SYSCR10)—H'FF25

Bit	7	6	5	4	3	2	1	0
	$IOF2_7$E	$IOF2_6$E	$IOF2_5$E	$IOF2_4$E	$IOF2_3$E	$IOF2_2$E	$IOF2_1$E	$IOF2_0$E
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

SYSCR10 selects the functions of the port 10 pins.

At reset and in the hardware standby mode, SYSCR10 is initialized to H'00. It is not initialized in the software standby mode.

Bit 7—IOF2$_7$ Enable (IOF2$_7$E): This bit selects the function of the P10$_7$ pin.

Bit 7 IOF2$_7$E	Description
0	P10$_7$ functions as an I/O pin. (Also, the initial value.)
1	P10$_7$ functions as an I/O pin of the ISP (IOF2$_7$).

Bit 6—IOF2$_6$ Enable (IOF2$_6$E): This bit selects the function of the P10$_6$ pin.

Bit 6 IOF2$_6$E	Description
0	P10$_6$ functions as an I/O pin. (Also, the initial value.)
1	P10$_6$ functions as an I/O pin of the ISP (IOF2$_6$).

Bit 5—IOF2$_5$ Enable (IOF2$_5$E): This bit selects the function of the P10$_5$ pin.

Bit 5 IOF2$_5$E	Description
0	P10$_5$ functions as an I/O pin. (Also, the initial value.)
1	P10$_5$ functions as an I/O pin of the ISP (IOF2$_5$).

Bit 4—IOF2$_4$ Enable (IOF2$_4$E): This bit selects the function of the P10$_4$ pin.

Bit 4 IOF2$_4$E	Description
0	P10$_4$ functions as an I/O pin. (Also, the initial value.)
1	P10$_4$ functions as an I/O pin of the ISP (IOF2$_4$).

Bit 3—IOF2$_3$ Enable (IOF2$_3$E): This bit selects the function of the P10$_3$ pin.

Bit 3 IOF2$_3$E	Description
0	P10$_3$ functions as an I/O pin. (Also, the initial value.)
1	P10$_3$ functions as an I/O pin of the ISP (IOF2$_3$).

Bit 2—IOF2$_2$ Enable (IOF2$_2$E): This bit selects the function of the P10$_2$ pin.

Bit 2 IOF2$_2$E	Description
0	P10$_2$ functions as an I/O pin. (Also, the initial value.)
1	P10$_2$ functions as an I/O pin of the ISP (IOF2$_2$).

Bit 1—IOF2$_1$ Enable (IOF2$_1$E): This bit selects the function of the P10$_1$ pin.

Bit 1 IOF2$_1$E	Description
0	P10$_1$ functions as an I/O pin. (Also, the initial value.)
1	P10$_1$ functions as an I/O pin of the ISP (IOF2$_1$).

Bit 0—IOF2$_0$ Enable (IOF2$_0$E): This bit selects the function of the P10$_0$ pin.

Bit 0 IOF2$_0$E	Description
0	P10$_0$ functions as an I/O pin. (Also, the initial value.)
1	P10$_0$ functions as an I/O pin of the ISP (IOF2$_0$).

9.9 Port 11

9.9.1 Overview

Port 11 is an 8-bit input/output port (figure 9.8). The pin functions are the same in all MCU operating modes.

Outputs from port 11 can drive one TTL load and a 30-pF capacitive load.

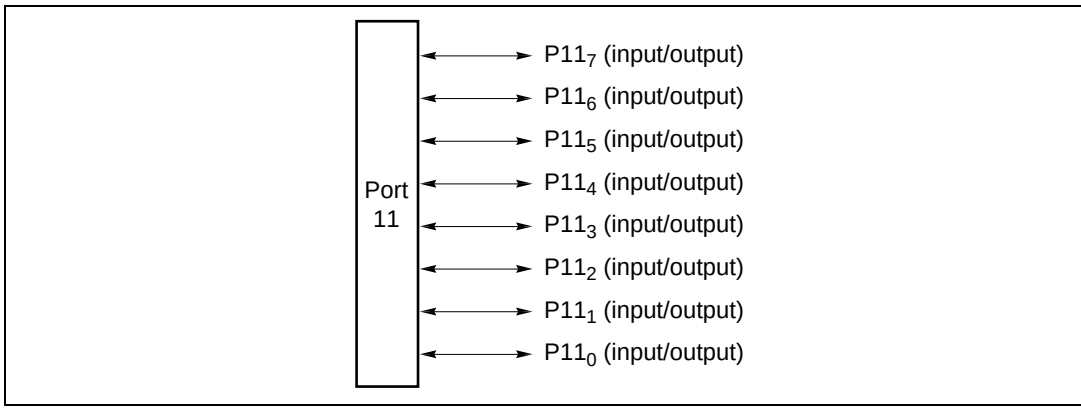


Figure 9.8 Pin Functions of Port 11

9.9.2 Port 11 Registers

Table 9.15 Configuration of Port 11 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 11 data direction register	P11DDR	W	H'00	H'FF36
Port 11 data register	P11DR	R/W	H'00	H'FE96

Port 11 Data Direction Register (P11DDR)—H'FF36

Bit	7	6	5	4	3	2	1	0
	$P11_7$DDR	$P11_6$DDR	$P11_5$DDR	$P11_4$DDR	$P11_3$DDR	$P11_2$DDR	$P11_1$DDR	$P11_0$DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

P11DDR is an 8-bit register that selects the direction of each pin in port 11. A pin functions as an output pin if the corresponding bit in P11DDR is set to 1, and as an input pin if the bit is cleared to 0.

P11DDR can only be written to. An attempt to read this register does not cause an error, but all bits will be read as 1, regardless of their true values.

At reset and in the hardware standby mode, P11DDR is initialized to H'00, setting all pins for input. P11DDR is not initialized in the software standby mode, so if a P11DDR bit is set to 1 when the MCU enters the software standby mode, the corresponding pin continues to output the value in the port 11 data register.

Port 11 Data Register (P11DR)—H'FE96

Bit	7	6	5	4	3	2	1	0
	$P11_7$	$P11_6$	$P11_5$	$P11_4$	$P11_3$	$P11_2$	$P11_1$	$P11_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

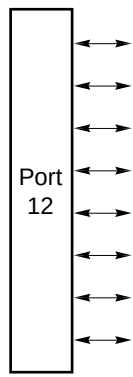
P11DR is an 8-bit register containing the data for pins $P11_7$ to $P11_0$. When the CPU reads P11DR, it reads the value in the P11DR latch for output pins, but for input pins, it obtains the pin status directly.

9.10 Port 12

9.10.1 Overview

Port 12 is an 8-bit input/output port (figure 9.9). It functions as an input/output port in modes 4 and 6. In modes 1, 3, and 5, it is used as D_{15}–D_8 of the data bus.

Outputs from port 12 can drive one TTL load and a 30-pF capacitive load.



Pin	For modes 4 and 6	For modes 1, 3, and 5
$P12_7/D_{15}$	$P12_7$ (input/output)	D_{15} (input/output)
$P12_6/D_{14}$	$P12_6$ (input/output)	D_{14} (input/output)
$P12_5/D_{13}$	$P12_5$ (input/output)	D_{13} (input/output)
$P12_4/D_{12}$	$P12_4$ (input/output)	D_{12} (input/output)
$P12_3/D_{11}$	$P12_3$ (input/output)	D_{11} (input/output)
$P12_2/D_{10}$	$P12_2$ (input/output)	D_{10} (input/output)
$P12_1/D_9$	$P12_1$ (input/output)	D_9 (input/output)
$P12_0/D_8$	$P12_0$ (input/output)	D_8 (input/output)

Figure 9.9 Pin Functions of Port 12

9.10.2 Port 12 Registers

Table 9.16 Configuration of Port 12 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 12 data direction register	P12DDR	W	H'00	H'FF37
Port 12 data register	P12DR	R/W	H'00	H'FE97

Port 12 Data Direction Register (P12DDR)—H'FF37

Bit	7	6	5	4	3	2	1	0
	$P12_7$DDR	$P12_6$DDR	$P12_5$DDR	$P12_4$DDR	$P12_3$DDR	$P12_2$DDR	$P12_1$DDR	$P12_0$DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

P12DDR is an 8-bit register that selects the direction of each pin in port 12. A pin functions as an output pin if the corresponding bit in P12DDR is set to 1, and as an input pin if the bit is cleared to 0.

P12DDR can only be written to. An attempt to read this register does not cause an error, but all bits will be read as 1, regardless of their true values.

At reset and in the hardware standby mode, P12DDR is initialized to H'00, setting all pins for input. P12DDR is not initialized in the software standby mode, so if a P12DDR bit is set to 1 when the MCU enters the software standby mode, the corresponding pin continues to output the value in the port 12 data register.

Port 12 Data Register (P12DR)—H'FE97

Bit	7	6	5	4	3	2	1	0
	$P12_7$	$P12_6$	$P12_5$	$P12_4$	$P12_3$	$P12_2$	$P12_1$	$P12_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P12DR is an 8-bit register containing the data for pins $P12_7$ to $P12_0$. When the CPU reads P12DR, it reads the value in the P12DR latch for output pins, but for input pins, it obtains the pin status directly.

9.10.3 Pin Functions in Each Mode

Port 12 has two different types of functions for modes 4 and 6, and modes 1, 3, and 5.

Pin Functions in Modes 4 and 6: In modes 4 and 6, each of the port 12 pins can be designated as an input pin or an output pin (figure 9.10) by setting the corresponding bit in P12DDR to 1 for output or clearing it to 0 for input.

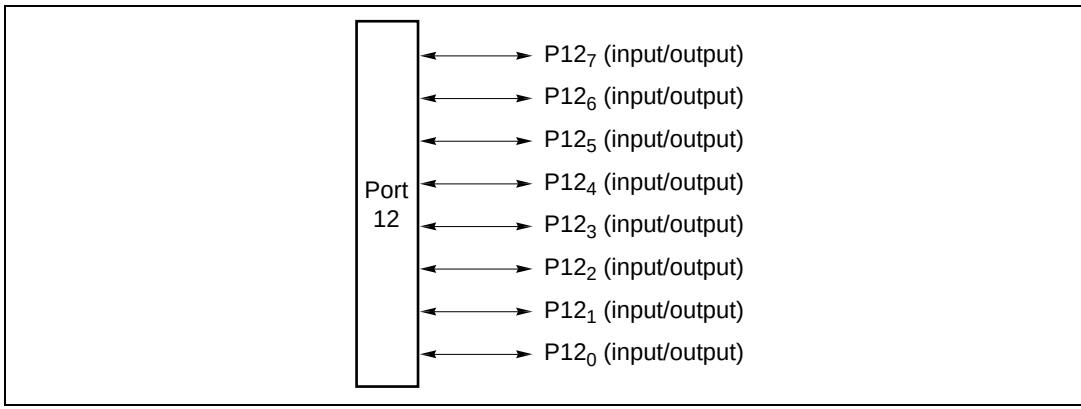


Figure 9.10 Port 12 Pin Functions in Modes 4 and 6

Pin Functions in Modes 1, 3, and 5: In modes 1, 3, and 5, port 12 is automatically used as the data bus and P12DDR is ignored.

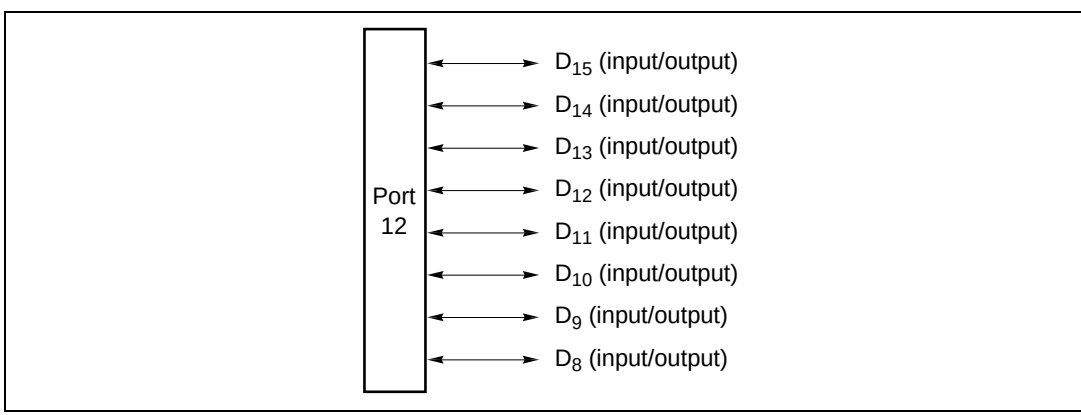


Figure 9.11 Port 12 Pin Functions in Modes 1, 3, and 5

Section 10 PWM (Pulse Width Modulation) Timer

10.1 Overview

The H8/570 MCU includes a one-channel pulse-width modulation (PWM) timer based on a 16-bit timer (TMR). The PWM timer can output six independent waveforms.

10.1.1 Features

- Selection of four clock sources: The 16-bit timer can be driven by an internal clock source (ø, ø/4, ø/8, or ø/16). If ø is selected, the pulse width is 100 ns for an oscillator frequency of 10 MHz.
- Three output compare registers
- Six output pins: Each output pin can independently generate pulses if the contents of the 16-bit timer and the three output compare registers match.
- Selection of two output modes
  - Parallel output mode: Direct output of a six-bit code.
  - Discrete output mode: Output of one bit per output compare register.
- Setting of timer count cycle: The timer control register (TCR) selects the timer count cycle of the 16-bit timer. The 16-bit timer can function as a free-running timer, or can be cleared by matching its contents with an output compare register.
- Three types of interrupts: These three maskable interrupts are generated if the contents of the 16-bit timer and an output compare register match. All interrupts can start a DTC (data transfer controller) data transfer.
- Two ISP transfer requests: Matching values between the 16-bit timer and an output compare register can set ISP flags. The ISP's transfer function can write data to the PWM registers.

10.1.2 Block Diagram

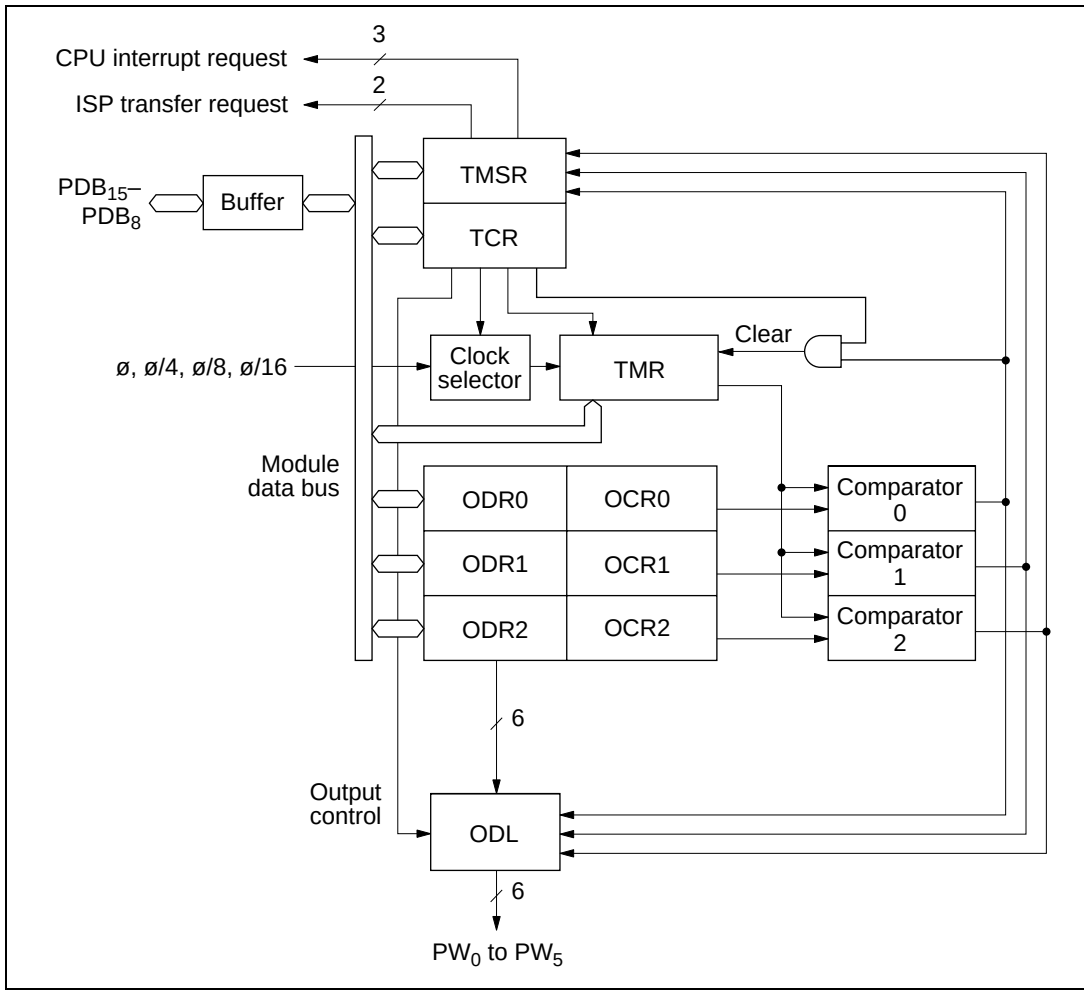


Figure 10.1 Block Diagram of PWM Timer

10.1.3 Output Pins

Name	Abbreviation	I/O	Function
PWM output pin 0	PW_0	Output	PWM output 0
PWM output pin 1	PW_1	Output	PWM output 1
PWM output pin 2	PW_2	Output	PWM output 2
PWM output pin 3	PW_3	Output	PWM output 3
PWM output pin 4	PW_4	Output	PWM output 4
PWM output pin 5	PW_5	Output	PWM output 5

10.1.4 Register Configuration

Name	Abbreviation	R/W	Initial Value	Address
Timer control register	TCR	R/W	H'00	H'FEA0
Timer status register	TMSR	R/W	H'00	H'FEA1
Output data latch	ODL	R/W	H'C0*	H'FEA2
Output data register 0	ODR0	R/W	H'C0*	H'FEA3
Output data register 1	ODR1	R/W	H'C0*	H'FEA4
Output data register 2	ODR2	R/W	H'C0*	H'FEA5
Output compare register 0 (High)	OCR0H	R/W	H'FF	H'FEA6
Output compare register 0 (Low)	OCR0L	R/W	H'FF	H'FEA7
Output compare register 1 (High)	OCR1H	R/W	H'FF	H'FEA8
Output compare register 1 (Low)	OCR1L	R/W	H'FF	H'FEA9
Output compare register 2 (High)	OCR2H	R/W	H'FF	H'FEAA
Output compare register 2 (Low)	OCR2L	R/W	H'FF	H'FEAB
Timer (High)	TMRH	R/W	H'00	H'FEAC
Timer (Low)	TMRL	R/W	H'00	H'FEAD

Note: * The initial value of unused bits is 1.

10.2 Register Descriptions

10.2.1 Timer (TMR)—H'FEAC, H'FEAD

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TMR is a 16-bit read/write up-counter that increments on an internal pulse generated from a clock source. The clock source is selected as ø, ø/4, ø/8, or ø/16 by the clock select 1 and 0 bits (CKS1 and CKS0) of the timer control register (TCR).

Under the control of the timer control register (TCR), TMR can function as a free-running timer, or can be cleared by containing the identical contents of an output compare register.

Because TMR is a 16-bit register, a temporary register (TEMP) is used when TMR is written to or read from. See section 10.3, PWM Operating Timing, for details.

TMR is initialized to H'0000 at reset and in the standby modes.

10.2.2 Output Compare Registers 0 to 2 (OCR0 to OCR2)—H'FEA6 and H'FEA7, H'FEA8 and H'FEA9, H'FEAA and H'FEAB

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

OCR0 to OCR2 are 16-bit read/write registers which hold the contents that are continually compared with the value in TMR. When a match is detected, a value in the output data register (ODR) corresponding to each OCR number is transferred to an output data latch (ODL).

In addition, if the output compare flag (OCF) and the output compare interrupt enable bit (OCIE) corresponding to each OCR are set to 1, an interrupt is requested to the CPU. A data transfer request can be output to the ISP. See section 10.2.6, Timer Status Register (TMSR), for details.

If the free-running mode bit ($\overline{\text{FRM}}$) in the timer control register (TCR) is set to 1, TMR is cleared by a match with OCR0. In this case, the value in OCR0 is transferred to ODL.

Because OCR0 to OCR2 are 16-bit registers, a temporary register (TEMP) is used when they are written to. See section 10.3, PWM Operating Timing, for details.

OCR0 to OCR2 are initialized to H'FFFF at reset and in the standby modes.

Note: The data transfer from an output data register (ODR) to the output data latch (ODL) and TMR clear are synchronized with the next clock timing at a match between TMR and OCR. For example, if N is set to OCR, data in ODR is transferred when the TMR value changes from N to N + 1. When TMR is cleared at the match with OCR0 and if the OCR0 value is N, TMR counts up from 0 to N. The counter cycle in this case is N + 1.

10.2.3 Output Data Registers 0 to 2 (ODR0 to ODR2)—H'FEA3, H'FEA4, H'FEA5

Bit	7	6	5	4	3	2	1	0
ODR2	—	—	OD25	OD24	OD23	OD22	OD21	OD20
ODR1	—	—	OD15	OD14	OD13	OD12	OD11	OD10
ODR0	—	—	OD05	OD04	OD03	OD02	OD01	OD00
Initial value	1	1	0	0	0	0	0	0
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W

ODR0 to ODR2 are 6-bit registers which temporarily store the output code for transferring to the output data latch (ODL). The CPU or the ISP control the setting of these registers.

Each register number corresponds to the OCR number. When a match exists between TMR and OCR, the value in ODR of the same number (0 to 2) as OCR is transferred to ODL. If the output mode select bit (OMS) in the timer control register (TCR) is set to 1, six bits of ODR is output in parallel. If OMS is cleared to 0, only one bit is output discretely. See section 10.2.5, Timer Control Register (TCR), for details.

10.2.4 Output Data Latch (ODL)—H'FEA2

Bit	7	6	5	4	3	2	1	0
	—	—	ODL5	ODL4	ODL3	ODL2	ODL1	ODL0
Initial value	1	1	0	0	0	0	0	0
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W

ODL is a 6-bit register which stores the PWM output value.

ODL usually latches the code output from ODR0 to ODR2 at the time of a match between TMR and OCR0 to OCR2. The CPU and the ISP can read from or write to ODL, and can directly set the initial value. See section 10.2.5, Timer Control Register (TCR), for details.

10.2.5 Timer Control Register (TCR)—H'FEA0

Bit	7	6	5	4	3	2	1	0
	TCE	$\overline{\text{FRM}}$	CKS1	CKS0	OMS	OE2	OE1	OE0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TCR is an 8-bit read/write register that selects the TMR clock source and the operating mode.

Bit 7—Timer Count Enable (TCE): This bit selects TMR to either count up or stop.

Bit 7 TCE	Description
0	TMR stops counting. (Also, the initial value.)
1	TMR counts up.

Bit 6—Free-Running Mode ($\overline{\text{FRM}}$): This bit selects the TMR count cycle.

Bit 6 $\overline{\text{FRM}}$	Description
0	TMR functions as a free-running timer counting from H'0000 to H'FFFF. (Also, the initial value.)
1	At a match between TMR and OCR0, TMR is cleared to H'0000 at the next clock cycle.

Bits 5 and 4—Clock Select 1 and 0 (CKS1 and CKS0): These bits select the internal clock source for TMR.

Bit 5 CKS1	Bit 4 CKS0	Description
0	0	Internal clock source (ø) (Also, the initial value.)
	1	Internal clock source (ø/4)
1	0	Internal clock source (ø/8)
	1	Internal clock source (ø/16)

Bit 3—Output Mode Select (OMS): This bit selects whether to transfer output codes in ODR0 to ODR2 to ODL in parallel or discretely. Output codes in the ODRs are transferred to ODL as follows.

Bit 3 OMS	ODL5	ODL4	ODL3	ODL2	ODL1	ODL0	Transfer Condition
0	—	—	—	—	—	OD00	TMR = OCR0 (Also, the initial value.)
	—	—	—	—	OD11	—	TMR = OCR1
	—	—	—	OD22	—	—	TMR = OCR2
1	OD05	OD04	OD03	OD02	OD01	OD00	TMR = OCR0
	OD15	OD14	OD13	OD12	OD11	OD10	TMR = OCR1
	OD25	OD24	OD23	OD22	OD21	OD20	TMR = OCR2

Note: A dash indicates that the contents in ODL do not change.

If a match occurs between TMR and two or more OCRs simultaneously when the OMS bit is set to 1, the priority for transfer conditions is followed as OCR0 > OCR1 > OCR2.

Bits 2 to 0—Output Enable 2 to 0 (OE2 to OE0): These bits select the PWM output pins. The PWM outputs depend on these bits and the OMS bit. The DDRs of pins selected for the PWM outputs are forcibly set, however other pins are not controlled by the PWM.

Bit 3 OMS	Bit 2 OE2	Bit 1 OE1	Bit 0 OE0	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0
0	0	0	0	—	—	—	—	—	— (Initial value)
			1	—	—	—	—	—	Output
		1	0	—	—	—	—	Output	Output
			1	—	—	—	Output	Output	Output
	1	*	*	—	—	—	Output	Output	Output
1	0	0	0	—	—	—	—	—	—
			1	—	—	—	—	—	Output
		1	0	—	—	—	—	Output	Output
			1	—	—	—	Output	Output	Output
	1	0	0	—	—	Output	Output	Output	Output
			1	—	Output	Output	Output	Output	Output
		1	*	Output	Output	Output	Output	Output	Output

Notes: * 0 or 1.
A dash indicates not being controlled by the PWM.

10.2.6 Timer Status Register (TMSR)—H'FEA1

Bit	7	6	5	4	3	2	1	0
	TRE2	TRE0	OCIE2	OCIE1	OCIE0	OCF2	OCF1	OCF0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/(W)*	R/(W)*	R/(W)*

TMSR is an 8-bit read and partially write* register that enables interrupts requested to the CPU and data transfer requested to the ISP.

TMSR is initialized to H'00 at reset and in the standby modes.

Note: Partially write implies that only 0s are allowed to be written in bits 2 to 0 by software to clear the flags.

Bit 7—Transfer Request Enable 2 (TRE2): This bit selects whether or not to request data transfer to the ISP.

Bit 7 TRE2	Description
0	The PWM cannot set interconnection flag high 7 (ICFH7) in the ISP. (Also, the initial value.)
1	The ICFH7 is set to 1 when output compare flag 2 (OCF2) is set to 1.

Bit 6—Transfer Request Enable 0 (TRE0): This bit selects whether or not to request data transfer to the ISP.

Bit 6 TRE0	Description
0	The PWM cannot set interconnection flag high 6 (ICFH6) in the ISP. (Also, the initial value.)
1	The ICFH6 is set to 1 when output compare flag 0 (OCF0) is set to 1.

Bit 5—Output Compare Interrupt Enable 2 (OCIE2): This bit selects whether or not to request output compare interrupt 2 (OCI2) when output compare flag 2 (OCF2) in the timer status register (TMSR) is set to 1.

Bit 5 OCIE2	Description
0	Output compare interrupt request 2 (OCI2) is disabled. (Also, the initial value.)
1	Output compare interrupt request 2 (OCI2) is enabled.

Bit 4—Output Compare Interrupt Enable 1 (OCIE1): This bit selects whether or not to request output compare interrupt 1 (OCI1) when output compare flag 1 (OCF1) in the timer status register (TMSR) is set to 1.

Bit 4 OCIE1	Description
0	Output compare interrupt request 1 (OCI1) is disabled. (Also, the initial value.)
1	Output compare interrupt request 1 (OCI1) is enabled.

Bit 3—Output Compare Interrupt Enable 0 (OCIE0): This bit selects whether or not to request output compare interrupt 0 (OCI0) when output compare flag 0 (OCF0) in the timer status register (TMSR) is set to 1.

Bit 3 OCIE0	Description
0	Output compare interrupt request 0 (OCI0) is disabled. (Also, the initial value.)
1	Output compare interrupt request 0 (OCI0) is enabled.

Bit 2—Output Compare Flag 2 (OCF2): This status flag is set to 1 when the TMR value matches the OCR2 value and the next count clock is input to TMR.

Bit 2 OCF2	Description
0	This bit is cleared to 0 when: • The CPU reads the OCF2 bit, then writes a 0 in this bit (The BCLR instruction can also clear this bit.) • The data transfer controller (DTC) is activated by the setting of OCF2 • Reset or by entering the standby modes (Also, the initial value.)
1	This bit is set to 1 when TMR = OCR2 and the next count clock is input to TMR.

Bit 1—Output Compare Flag 1 (OCF1): This status flag is set to 1 when the TMR value matches the OCR1 value and the next count clock is input to TMR.

Bit 1 OCF1	Description
0	This bit is cleared to 0 when: • The CPU reads the OCF1 bit, then writes a 0 in this bit (The BCLR instruction can also clear this bit.) • The data transfer controller (DTC) is activated by the setting of OCF1 • Reset or by entering the standby modes (Also, the initial value.)
1	This bit is set to 1 when TMR = OCR1 and the next count clock is input to TMR.

Bit 0—Output Compare Flag 0 (OCF0): This status flag is set to 1 when the TMR value matches the OCR0 value and the next count clock is input to TMR.

Bit 0 OCF0	Description
0	This bit is cleared to 0 when: • The CPU reads the OCF0 bit, then writes a 0 in this bit (The BCLR instruction can also clear this bit.) • The data transfer controller (DTC) is activated by the setting of OCF0 • Reset or by entering the standby modes (Also, the initial value.)
1	This bit is set to 1 when TMR = OCR0 and the next count clock is input to TMR.

10.3 PWM Operating Timing

The timing of PWM outputs is based on the matching between TMR and three OCRs. If a match between TMR and an OCR occurs, a match-acknowledge signal is output from a comparator and the PWM operates as follows.

1. Following the output mode which is set in TCR, corresponding ODR values are transferred to ODL. The ODL value is then directly output to the external pins.
2. A corresponding output compare flag in TMSR is set to 1. Following the setting of TMSR bits, a corresponding ISP flag is simultaneously set to 1.
3. If TMR and OCR0 values match when the $\overline{\text{FRM}}$ bit in TCR is set to 1, TMR is cleared to H'0000.

The match-acknowledge signal is actually generated at the next TMR count clock timing after the match, not during the comparison between TMR and OCR (figure 10.2). When $\overline{\text{FRM}}$ = 0, the figure remains the same except that TMR is not cleared. Pay particular attention to the relationship between the OCR set value and the count cycle or the pulse width when $\overline{\text{FRM}}$ = 1.

Output compare flags (OCF) and ISP flags are set to 1 during TMR incrementation or TMR clear after the match between TMR and OCR values, hence synchronizing with the timing of the match-acknowledge signal generation.

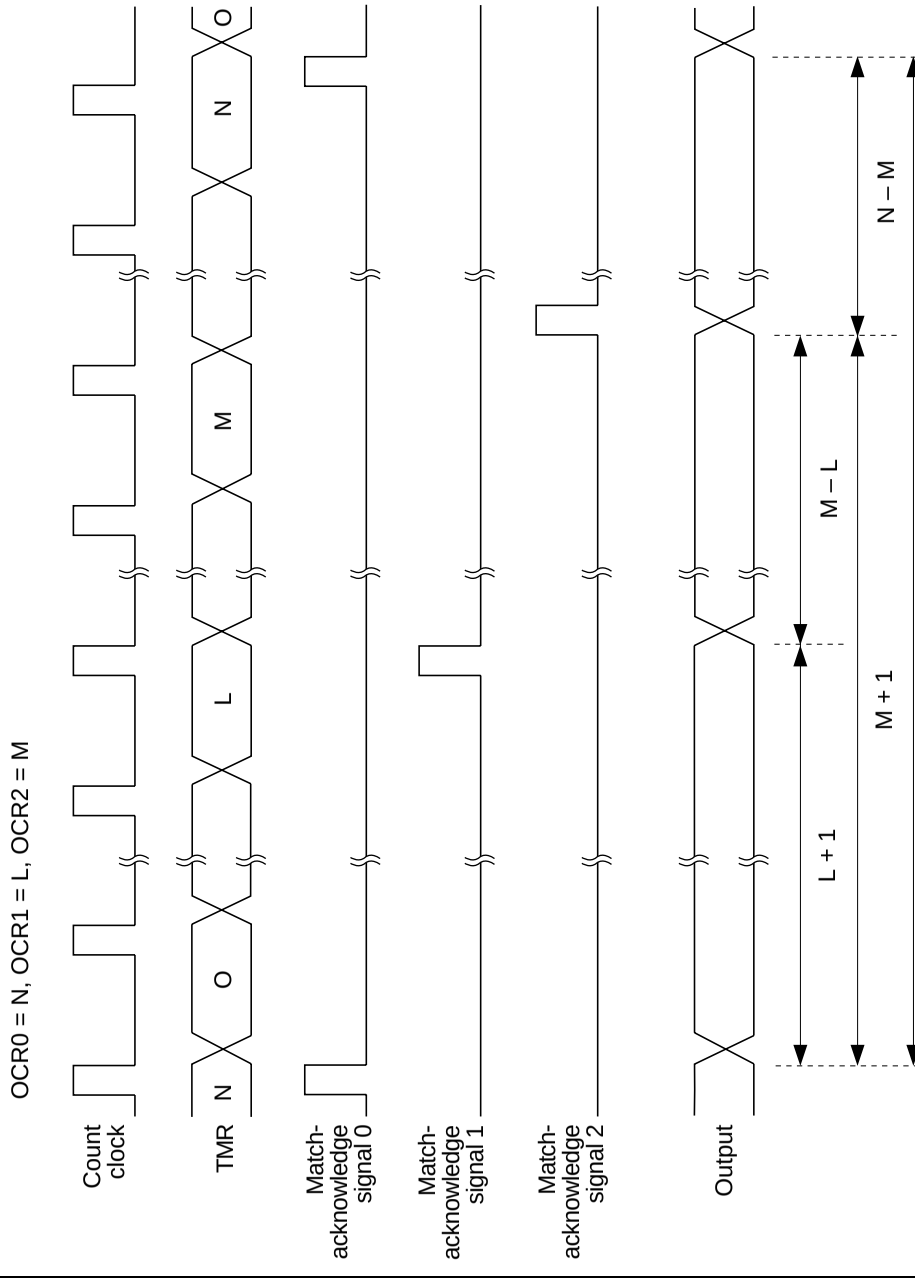


Figure 10.2 PWM Operating Timing

10.4 Interface with CPU and ISP

10.4.1 Read or Write of TMR and OCRs

Although TMR and OCR0 to OCR2 are 16-bit registers, they are connected to an 8-bit data bus. When the CPU accesses these registers, the access is performed using an 8-bit temporary register (TEMP) to ensure that both bytes are written or read simultaneously.

These registers are written and read as follows.

- TMR and OCR0 to OCR2 write

  When the CPU writes to the upper byte, the upper byte of the write data is placed in TEMP. Next, when the CPU writes to the lower byte, this byte of data is combined with the byte in TEMP and all 16 bits are written in the register simultaneously.

- TMR read

  When the CPU reads the upper byte, the upper byte of data is sent to the CPU and the lower byte is placed in TEMP. When the CPU reads the lower byte, it receives the value from TEMP.

Programs that access these registers should normally use word access. However, programs may first access the upper byte, then the lower byte. Data will not be transferred correctly if the bytes are accessed in reverse order, or if only one byte is accessed.

Coding Examples:

Write the contents of R0 into OCR2:	`MOV.W R0,@H'FEAA`
Read TMR:	`MOV.W @H'FEAC,R0`

When OCR0 to OCR2 are read, the upper and lower bytes are both transferred directly to the CPU without using the temporary register.

10.4.2 Write to TMR

The CPU and ISP can write to TMR during both the counting operation and the halt period. If the match-acknowledge signal between TMR and OCR0 is generated during the T_3 state of a write cycle to TMR, TMR clear takes priority and the write is not performed.

10.4.3 Write to OCRs

When the CPU or ISP writes to an OCR, the match-acknowledge signal between the concerned OCR and TMR is forcibly inhibited during one cycle of data write to OCR. The matching operation is not performed regardless of the OCR value before and after the write operation (figure 10.3). As for other OCRs, however, the matching operation is normally performed.

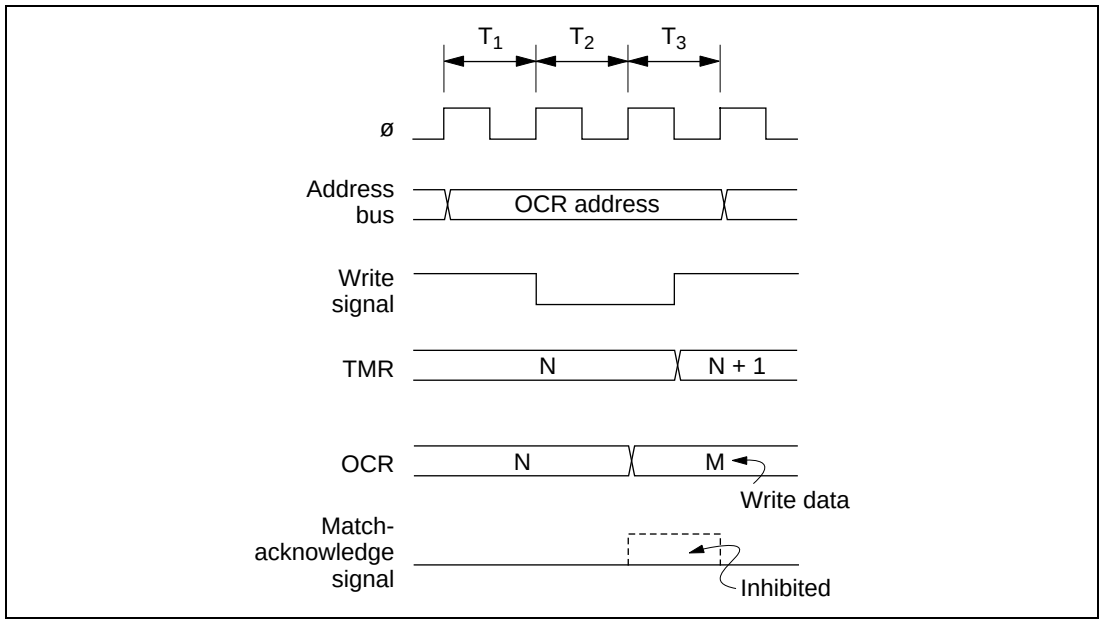


Figure 10.3 TMR Write-Clear Contention

10.5 CPU Interrupts and DTC Interrupts

The PWM timer can request three interrupts: OCI0, OCI1, and OCI2. Each of these interrupts (table 10.3) is requested when a match between the corresponding OCR and TMR occurs. The interrupt controller outputs independent interrupt vectors for each interrupt.

Table 10.1 PWM Timer Interrupts

Interrupt	Description	DTC Service Available?	Priority
OCI0	Requested when OCF0 is set	Yes	High
OCI1	Requested when OCF1 is set	Yes	↑
OCI2	Requested when OCF2 is set	Yes	Low

These interrupts can be directed to the data transfer controller (DTC) to enable a data transfer.

When the DTC serves one of these interrupts, it automatically clears the OCF2, OCF1, or OCF0 flag to 0. See section 6, Data Transfer Controller, for further information on the DTC.

10.6 Data Transfer by ISP

Setting data in PWM registers, especially to the three output compare registers (OCR) and the output data registers (ODR), can not only be done by CPU interrupts but also by the ISP memory transfer function. For data transfer by the ISP, the match-acknowledge signal between TMR and OCR0 or OCR2 can directly set an ISP flag (ICFH7 or ICFH6). The TRE2 and TRE0 bits in the timer status register (TMSR) select whether or not to set the ISP flag at a match in the PWM timer.

For example, by creating a table of PWM outputs and output timing in memory, the ISP can read the table and renew any OCR or ODR values each time a match occurs. Consequently, the overhead of the CPU decreases, and the high responsibility for data transfer requests is achieved. Moreover, the ISP can perform easy operations as required, in addition to performing a direct memory access (DMA) transfer of the table data.

10.7 Sample Application

10.7.1 Six-Phase Pulse Output

A six-phase pulse output can be used as an application for AC motor control. See figure 10.4. In this example, OCR0 is used to set the TMR count cycle. Timing of output changing points is set in OCR1 and OCR2, and the output codes are set in ODR1 and ODR2. Each time a data transfer is requested at a match between TMR and OCR2, the ISP can renew OCR and ODR values by reading the table in memory.

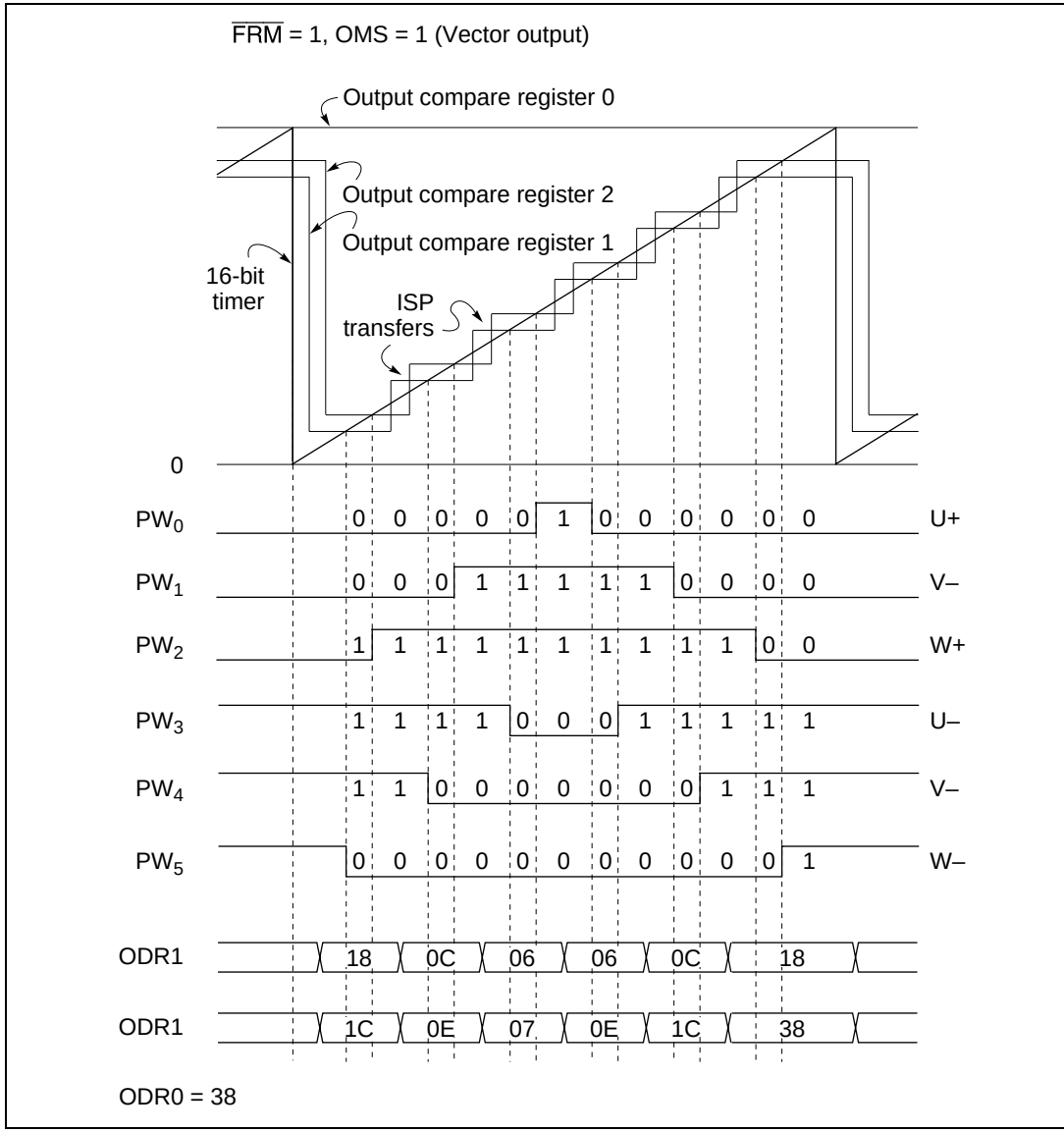

FRM = 1, OMS = 1 (Vector output)
Output compare register 0
Output compare register 2
Output compare register 1
16-bit timer
ISP transfers
0
PW0 0 0 0 0 0 1 0 0 0 0 0 0 U+
PW1 0 0 0 1 1 1 1 1 0 0 0 0 V–
PW2 1 1 1 1 1 1 1 1 1 1 0 0 W+
PW3 1 1 1 1 0 0 0 1 1 1 1 1 U–
PW4 1 1 0 0 0 0 0 0 0 1 1 1 V–
PW5 0 0 0 0 0 0 0 0 0 0 0 1 W–
ODR1 18 0C 06 06 0C 18
ODR1 1C 0E 07 0E 1C 38
ODR0 = 38

Figure 10.4 Waveform Output for AC Motor Control

10.7.2 Independent Waveform Output

Three independent waveforms are output if both the OMS and $\overline{\text{FRM}}$ bits are set to 0 (figure 10.5). TMR becomes a free-running timer. OCR0 and ODR0, OCR1 and ODR1, and OCR2 and ODR2 affect the output of pins PW_0, PW_1, and PW_2, respectively. OCR0 and OCR2 can request interrupts to the CPU and data transfer to the ISP, but OCR1 can only request interrupts to the CPU. In this example, the ISP can share the output control such as including the OCR values.

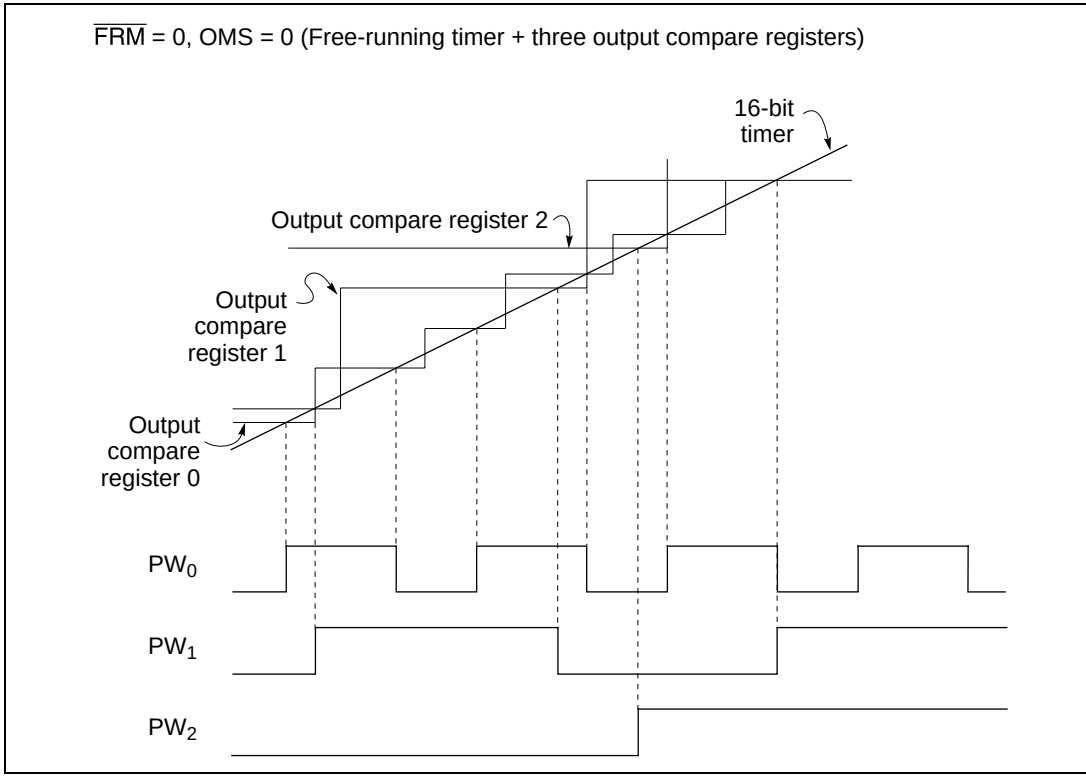


Figure 10.5 Independent Waveform Output

10.8 Application Notes

Changing the operating mode, clock source, and output select of TMR must be done during the TMR halt period (the period when the TCE bit in the timer control register (TCR) is set to 0). This is required since the expected output values are not guaranteed if they are changed during the TMR counting operation.

If the values in the TCE bit and the OMS bit in the timer control register (TCR) are simultaneously changed, the specification in the TCE bit may become ineffective.

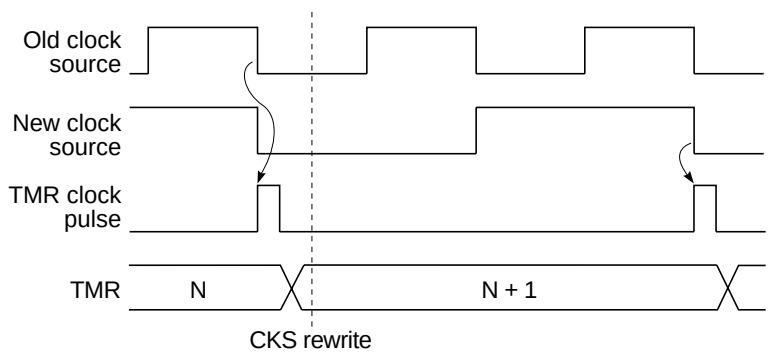
10.8.1 Incrementation Caused by Changing the Internal Clock Source

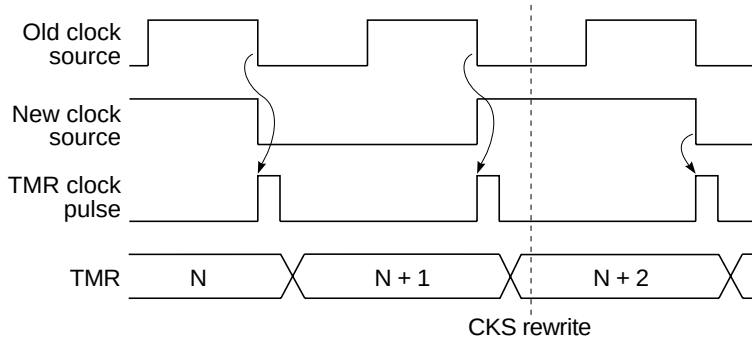
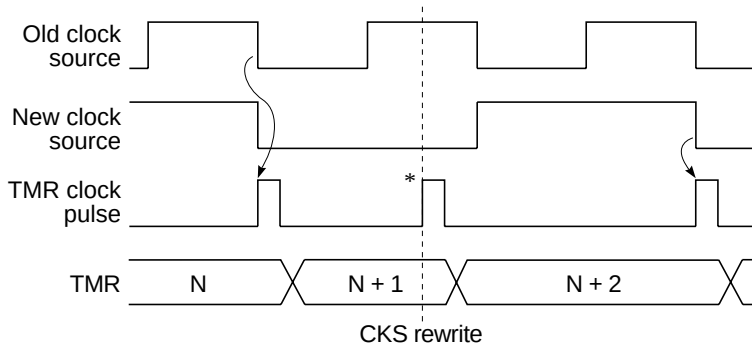
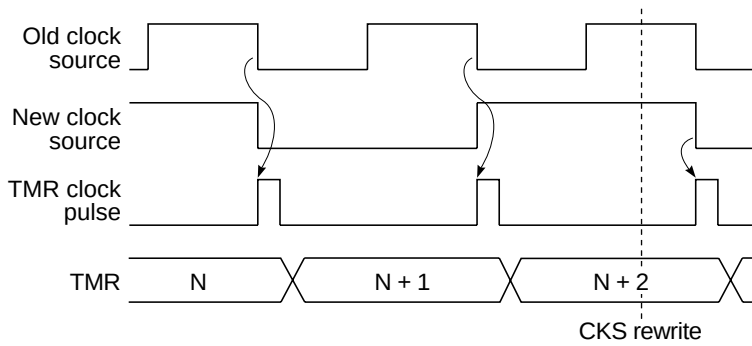
When the internal clock source is changed, the changeover may cause TMR to increment. This depends on the time at which the clock select bits (CKS1 and CKS0) are rewritten (table 10.2).

The pulse that increments TMR is generated at the falling edge of the internal clock source. If the clock source is changed when the old source is high and the new source is low, as in the high to low case in table 10.2, the changeover generates a falling edge that triggers the TMR clock pulse and increments TMR.

Switching between an internal and external clock source can also cause TMR to increment.

Table 10.2 Effect of Changing Internal Clock Sources

Description	Timing Chart
Low → Low: CKS1 and CKS0 are rewritten while both clock sources are low.	

Description	Timing Chart
Low → High: CKS1 and CKS0 are rewritten while the old clock source is low and the new clock source is high.	
High → Low: CKS1 and CKS0 are rewritten while the old clock source is high and the new clock source is low.	
High → High: CKS1 and CKS0 are rewritten while both clock sources are high.	

Note: * The switching of clock sources is regarded as a falling edge that increments TMR.

Section 11 Watchdog Timer

11.1 Overview

The H8/570 has an on-chip watchdog timer (WDT) module. This module can monitor system operation by requesting a reset signal if a system crash causes the timer count to overflow.

When this watchdog function is not needed, the WDT module can be used as an interval timer. In this interval timer mode, a WDT interrupt is requested at each counter overflow.

11.1.1 Features

- Selection of eight clock sources
- Selection of two modes: Watchdog timer mode or interval timer mode
- Counter overflow generates a reset signal in the watchdog timer mode or WDT interrupt request in the interval timer mode
- Reset signal generated by overflow can reset the H8/570 MCU, and can be output to external devices

11.1.2 Block Diagram

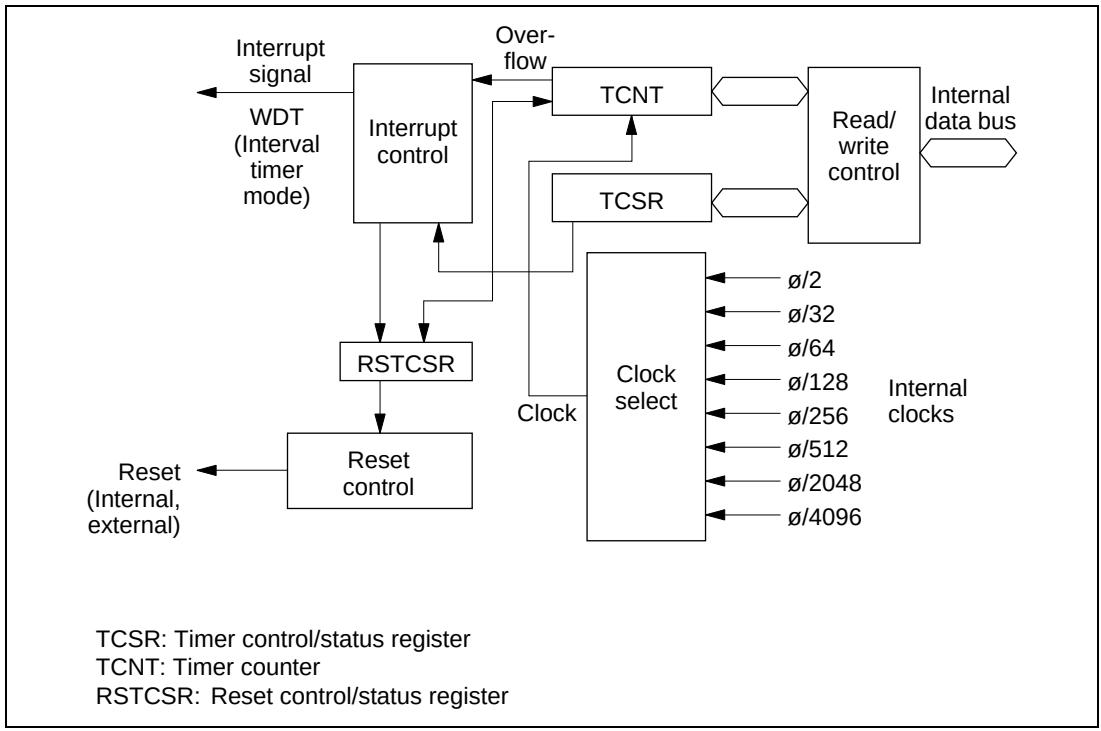


Figure 11.1 Block Diagram of Watchdog Timer

11.1.3 Register Configuration

Name	Abbreviation	R/W	Initial Value	Addresses	
				Write	Read
Timer control/status register	TCSR	R/(W)*	H'18	H'FE8B	H'FE8A
Timer counter	TCNT	R/W	H'00	H'FE8B	H'FE8B
Reset control/status register	RSTCSR	R/(W)*	H'3F	H'FF4F	H'FF4F

Note: * By software, only a 0 can be written to clear bit 7.

11.2 Register Descriptions

11.2.1 Timer Counter (TCNT)—H'FE8B

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The watchdog timer counter (TCNT) is a read/write* 8-bit up-counter. When the timer enable bit (TME) in the timer control/status register (TCSR) is set to 1, the timer counter starts counting the pulses of an internal clock source selected by clock select bits 2 to 0 (CKS2 to CKS0) in TCSR. When the count overflows (changes from H'FF to H'00), an interrupt is requested or a reset signal is generated.

The watchdog timer counter is initialized to H'00 at reset and when the TME bit is cleared to 0.

Note: * TCNT is write-protected by a password. See section 11.2.4, Notes on Register Access, for details.

11.2.2 Timer Control/Status Register (TCSR)—H'FE8A (Read), H'FE8B (Write)

Bit	7	6	5	4	3	2	1	0
	OVF	WT/$\overline{\text{IT}}$	TME	—	—	CKS2	CKS1	CKS0
Initial value	0	0	0	1	1	0	0	0
Read/Write	R/(W)*	R/W	R/W	—	—	R/W	R/W	R/W

Note: * By software, only a 0 can be written to clear the flag.

The watchdog timer control/status register (TCSR) is an 8-bit read/write* register that selects the timer mode and clock source, and performs other functions.

Bits 7 to 5 are initialized to 0 at reset and in the standby modes. Bits 2 to 0 are initialized to 0 at reset, but retain their values in the software standby mode.

Note: * The TCSR is write-protected by a password. See section 11.2.4, Notes on Register Access, for details.

Bit 7—Overflow Flag (OVF): This bit indicates that the watchdog timer count has overflowed.

Bit 7 OVF	Description
0	This bit is cleared to 0 after the CPU reads it then writes a 0 to it. (Also, the initial value.)
1	This bit is set to 1 when TCNT changes from H'FF to H'00 in the interval timer mode.*

Note: * The OVF is never set to 1 in the watchdog timer mode.

Bit 6—Timer Mode Select (WT/$\overline{IT}$): This bit selects whether to operate in the watchdog timer mode or interval timer mode.

Bit 6 WT/$\overline{IT}$	Description
0	Interval timer mode (WDT interrupt request). (Also, the initial value.)
1	Watchdog timer mode (reset output).

Bit 5—Timer Enable (TME): This bit enables or disables the timer.

Bit 5 TME	Description
0	TCNT is initialized to H'00 and stopped. (Also, the initial value.)
1	TCNT runs. A reset signal is generated or an interrupt is requested when the count overflows.

Bits 4 and 3—Reserved: These bits cannot be modified and are always read as 1.

Bits 2, 1, and 0—Clock Select (CKS2, CKS1, and CKS0): These bits select one of eight clock sources obtained by dividing the system clock (ø).

The overflow interval listed in the following table is the time from when the watchdog timer counter begins counting from H'00 until an overflow occurs.

Bit 2 CKS2	Bit 1 CKS1	Bit 0 CKS0	Description	
			Clock Source	Overflow Interval (ø = 10 MHz)
0	0	0	ø/2	51.2 µs (Also, the initial value.)
		1	ø/32	819.2 µs
	1	0	ø/64	1.6 ms
		1	ø/128	3.3 ms
1	0	0	ø/256	6.6 ms
		1	ø/512	13.1 ms
	1	0	ø/2048	52.4 ms
		1	ø/4096	104.9 ms

11.2.3 Reset Control/Status Register (RSTCSR)—H'FF4F

Bit	7	6	5	4	3	2	1	0
	WRST	RSTOE	—	—	—	—	—	—
Initial value	0	0	1	1	1	1	1	1
Read/Write	R/(W)*	R/W	—	—	—	—	—	—

Note: * By software, only a 0 can be written to clear the flag.

The reset control/status register (RSTCSR) is an 8-bit read/write* register that controls the reset signal generated by the watchdog timer overflow and the output of the reset signal to external devices.

Bit 6 is initialized to 0 by the reset signal input from the $\overline{\text{RES}}$ pin, but is not initialized by the reset signal generated by the watchdog timer overflow.

Note: * The RSTCSR is write-protected by a password. See section 11.2.4, Notes on Register Access, for details.

Bit 7—Watchdog Timer Reset (WRST): This bit indicates that the timer counter (TCNT) has overflowed in the watchdog timer mode, and a reset signal has been generated.

The reset signal generated by the overflow resets the H8/570 MCU. At the same time, if the RSTOE bit is set to 1, a low-level signal can be output from the $\overline{\text{RES}}$ pin to initialize the system.

The WRST bit is not cleared to 0 by the reset signal generated by the watchdog timer overflow, but is cleared when a reset signal is input from the $\overline{\text{RES}}$ pin, or when a 0 is written by the program in this bit after the CPU reads it while it is set to 1.

Bit 7 WRST	Description
0	This bit is cleared to 0 when a reset signal is input from the $\overline{\text{RES}}$ pin, or when a 0 is written by software in this bit after the CPU reads this bit while it is set to 1. (Also, the initial value.)
1	This bit is set to 1 when TCNT overflows in the watchdog timer mode and a reset signal is generated.

Bit 6—Reset Output Enable (RSTOE): This bit selects whether or not to output the reset signal generated by TCNT overflow in the watchdog timer mode to any external devices from the $\overline{\text{RES}}$ pin.

Bit 6 RSTOE	Description
0	The reset signal generated by TCNT overflow is not output to the external devices. (Also, the initial value.)
1	The reset signal generated by TCNT overflow is output to the external devices.

Bits 5 to 0—Reserved: These bits cannot be modified and are always read as 1.

11.2.4 Notes on Register Access

The watchdog timer registers TCNT, TCSR, and RSTCSR differ from other registers by being a bit more difficult to write to. The procedures for writing and reading these registers are given below.

Writing to TCNT and TCSR: These registers must be write accessed as words. Programs cannot write access them as bytes. The word must contain the write data and a password.

The watchdog timer registers TCNT and TCSR both have the same write address. The write data must be in the lower byte of the word written at this address. The upper byte must be either H'5A (password for TCNT) or H'A5 (password for TCSR). See figure 11.2.

The result of the access depicted in figure 11.2 is the transfer of the write data from the lower byte to the TCNT or TCSR.

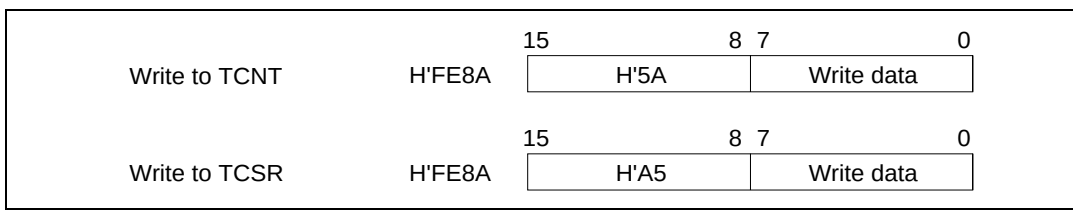


Figure 11.2 Writing to TCNT and TCSR

Coding Examples:

To clear TCNT to 00: `MOV.W #H'5A00,@H'FE8A`
To write H'4F in TCSR: `MOV.W #H'A54F,@H'FE8A`

Writing to RSTCSR: The RSTCSR is write accessed as words. The write address is H'FF4E. Programs cannot write access it as bytes.

Writing a 0 to the WRST bit differs from writing data to the RSTOE bit as follows.

For writing a 0 to the WRST bit, the upper byte contains H'A5 and the lower byte contains H'00 by word access. Consequently, the WRST bit is cleared to 0.

As for writing data to the RSTOE bit, the upper byte contains H'5A and the lower byte contains the write data by word access. Consequently, data in bit 6 of the lower byte is written in the RSTOE bit. See figure 11.3.

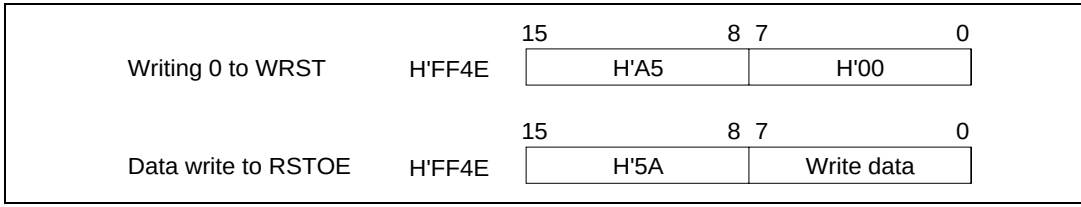


Figure 11.3 Writing to RSTCSR

Reading TCNT, TCSR, and RSTCSR: The read addresses are H'FE8A for TCSR, H'FE8B for TCNT, and H'FF4F for RSTCSR (table 11.1).

Reading these registers are read as other registers. Byte access instructions can be used.

Table 11.1 Read Addresses of TCNT, TCSR, and RSTCSR

Read Address	Register
H'FE8A	TCSR
H'FE8B	TCNT
H'FF4F	RSTCSR

11.3 Operation

11.3.1 Watchdog Timer Mode

The watchdog timer function begins operating when the WT/$\overline{IT}$ and TME bits are set to 1 in TCSR by software. Thereafter, the program should also periodically rewrite the contents of the timer counter (normally by writing H'00) to prevent the count from overflowing. If a program crash causes the timer count to overflow, the watchdog timer generates a reset signal (figure 11.4).

The reset signal generated by the watchdog timer can be output from the $\overline{RES}$ pin to the external devices to reset an external system. The reset signal is output for 516ø clock cycles. The RSTOE bit in the RSTCSR selects whether or not the signal is to be output to the external devices.

Since the reset signal from the watchdog timer has the same vector as the reset signal from the $\overline{RES}$ pin, the software must check the WRST bit in the RSTCSR to determine the source of the reset signal.

If a reset input from the $\overline{RES}$ pin and a reset by the watchdog timer overflow are simultaneously generated, the operation will differ depending on the RSTOE bit in the RSTCSR as follows.

If both reset signals are simultaneously generated when the RSTOE is set to 1, the reset signal generated by the watchdog timer takes priority. Therefore, the H8/570 MCU is reset, and the reset signal is output from the $\overline{RES}$ pin to the external devices for 516ø clock cycles. If the reset signal continues to be input from the $\overline{RES}$ pin after the 516ø clock cycle period, the signal becomes effective and the WRST and RSTOE bits are cleared to 0. If a reset is input from the $\overline{RES}$ pin before the watchdog timer overflows, that reset input becomes effective and the RSTOE bit is cleared to 0.

If both reset signals are simultaneously generated when the RSTOE is cleared to 0, the reset signal input from the $\overline{RES}$ pin takes priority and the WRST bit is cleared to 0.

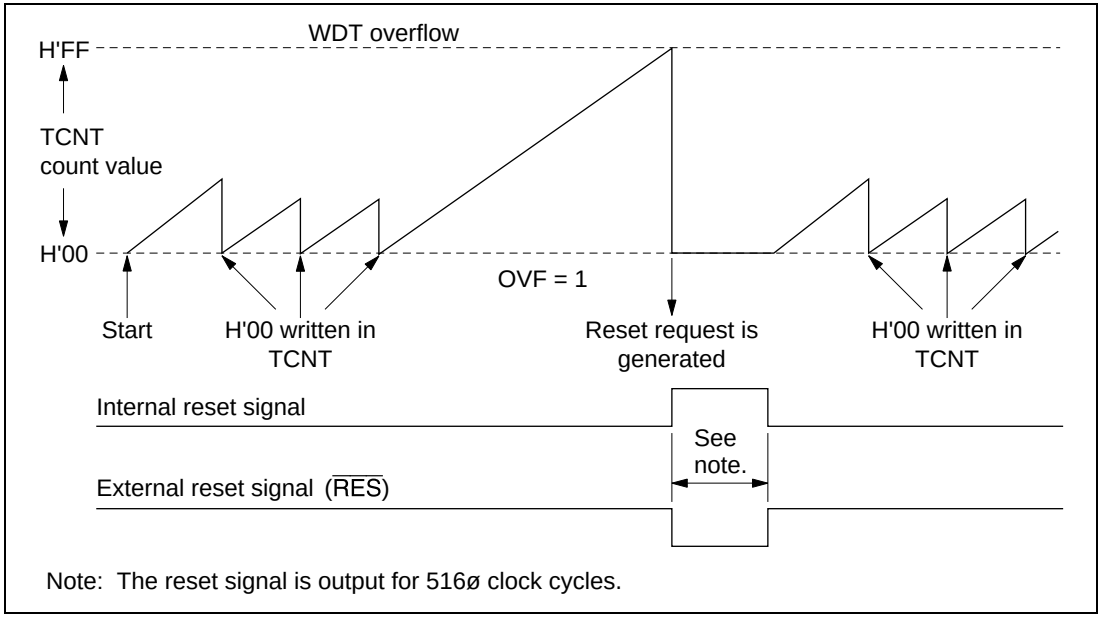


Figure 11.4 Operation in Watchdog Timer Mode

11.3.2 Interval Timer Mode

The interval timer operation begins when the WT/$\overline{IT}$ bit is cleared to 0 and the TME bit is set to 1.

In the interval timer mode, an WDT request is generated each time the timer count overflows. This function can be used to generate WDT requests at regular intervals. See figure 11.5.

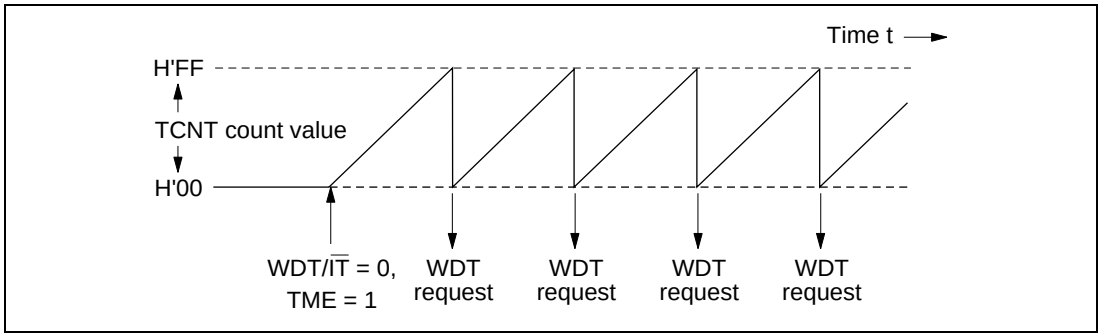


Figure 11.5 Operation in Interval Timer Mode

11.3.3 Operation in Software Standby Mode

Specific watchdog timer settings are required when using the software standby mode.

The TME bit must be cleared to 0 to stop the watchdog timer counter before a transition is made to the software standby mode. The MCU cannot enter the software standby mode while the TME bit is set to 1.

11.3.4 Setting of Overflow Flag

The OVF bit is set to 1 when the timer count overflows. Simultaneously, the WDT module requests a WDT interrupt. See figure 11.6.

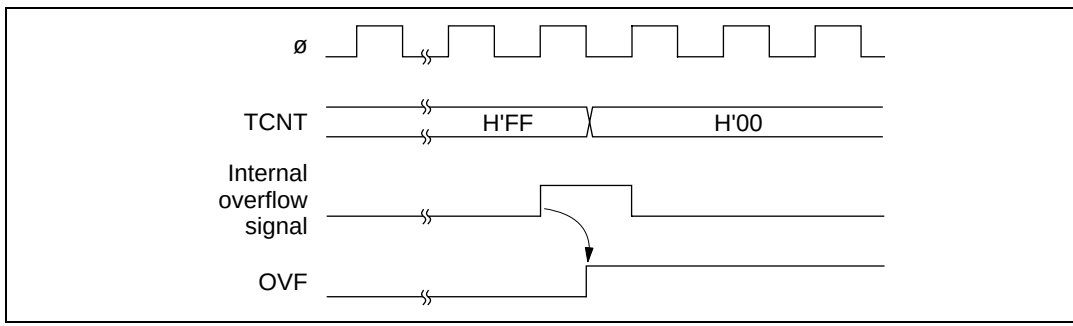


Figure 11.6 Setting of OVF Bit

11.3.5 Setting of Watchdog Timer Reset Bit

The WRST bit is effective when the WT/$\overline{IT}$ and TME bits are set to 1 in TCSR by software. The WRST bit is set to 1 when TCNT overflows. Simultaneously, a reset signal is generated to the H8/570 MCU (figure 11.7).

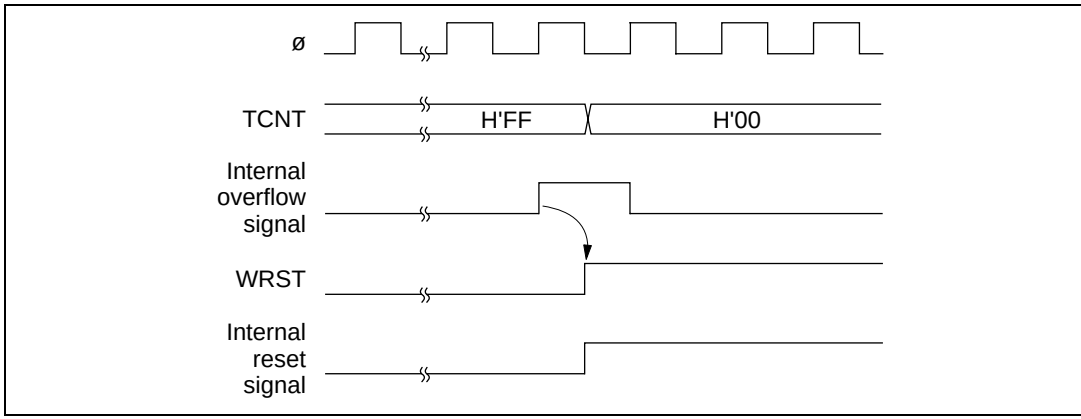


Figure 11.7 Setting of WRST Bit

11.4 Application Notes

Contention between TCNT Write and Increment: If a timer counter clock pulse is generated during the T_3 state of a write cycle to the timer counter, the write takes priority and the timer counter is not incremented. See figure 11.8.

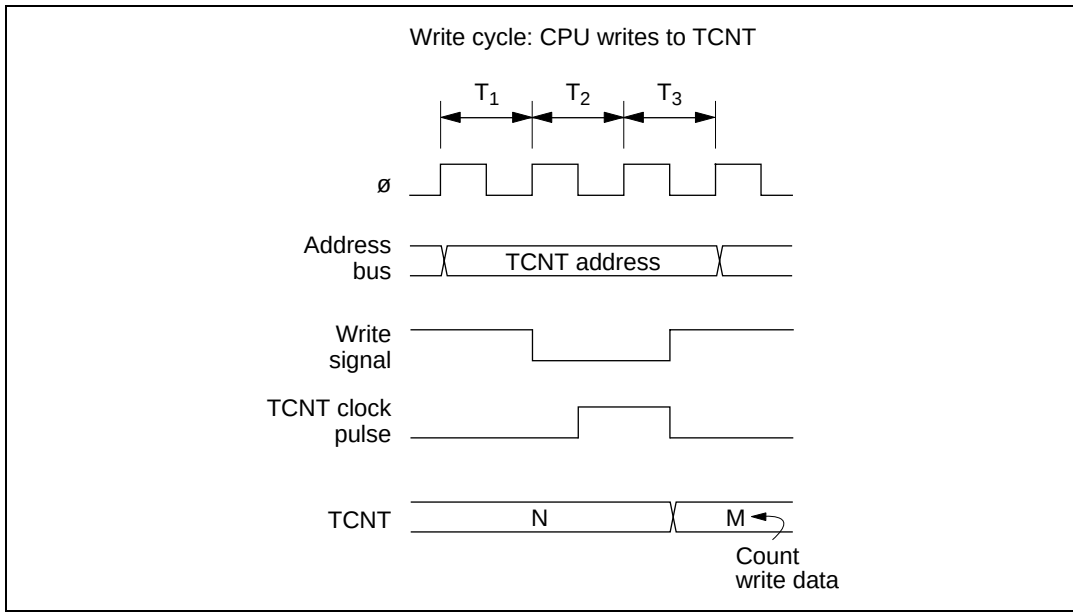


Figure 11.8 TCNT Write-Increment Contention

Changing the Clock Select Bits (CKS2–CKS0): The software should be programmed to stop the watchdog timer (by clearing the TME bit to 0) before changing the value of the clock select bits. If the clock select bits are modified while the watchdog timer is running, the timer count may be incremented incorrectly.

Section 12 Serial Communication Interface

12.1 Overview

The H8/570 MCU includes a single-channel serial communication interface (SCI) for transferring serial data to and from other devices. The SCI supports both synchronous and asynchronous data transfer. Communication control functions are provided by eight internal registers.

12.1.1 Features

- Selection of asynchronous or synchronous mode
  - — Asynchronous mode

    The SCI can communicate with a UART (Universal Asynchronous Receiver/Transmitter), ACIA (Asynchronous Communication Interface Adapter), or other devices that employ standard asynchronous serial communication. Eight data formats are available.

    Data length: 7 or 8 bits

    Stop bit length: 1 or 2 bits

    Parity: Even, odd, or none

    Error detection: Parity, overrun, and framing errors
  - — Synchronous mode

    The SCI can communicate with devices that synchronize data transfers with clock pulses.

    Data length: 8 bits

    Error detection: Overrun errors
- Full duplex communication

  The transmitting and receiving sections of the SCI are independent, so the SCI can transmit and receive simultaneously. Both the transmit and receive sections use double buffering, so continuous data transfer is possible in either direction.
- Built-in baud rate generator

  Any specified bit rate can be generated.
- Internal or external clock source

  The baud rate generator can operate on an internal clock source, or an external clock signal input at the SCK pin.
- Three interrupts

  Transmit-end, receive-end, and receive-error interrupts are requested independently. The transmit-end and receive-end interrupts can be served by the data transfer controller (DTC), providing a convenient way to transfer data with minimal CPU programming.

12.1.2 Block Diagram

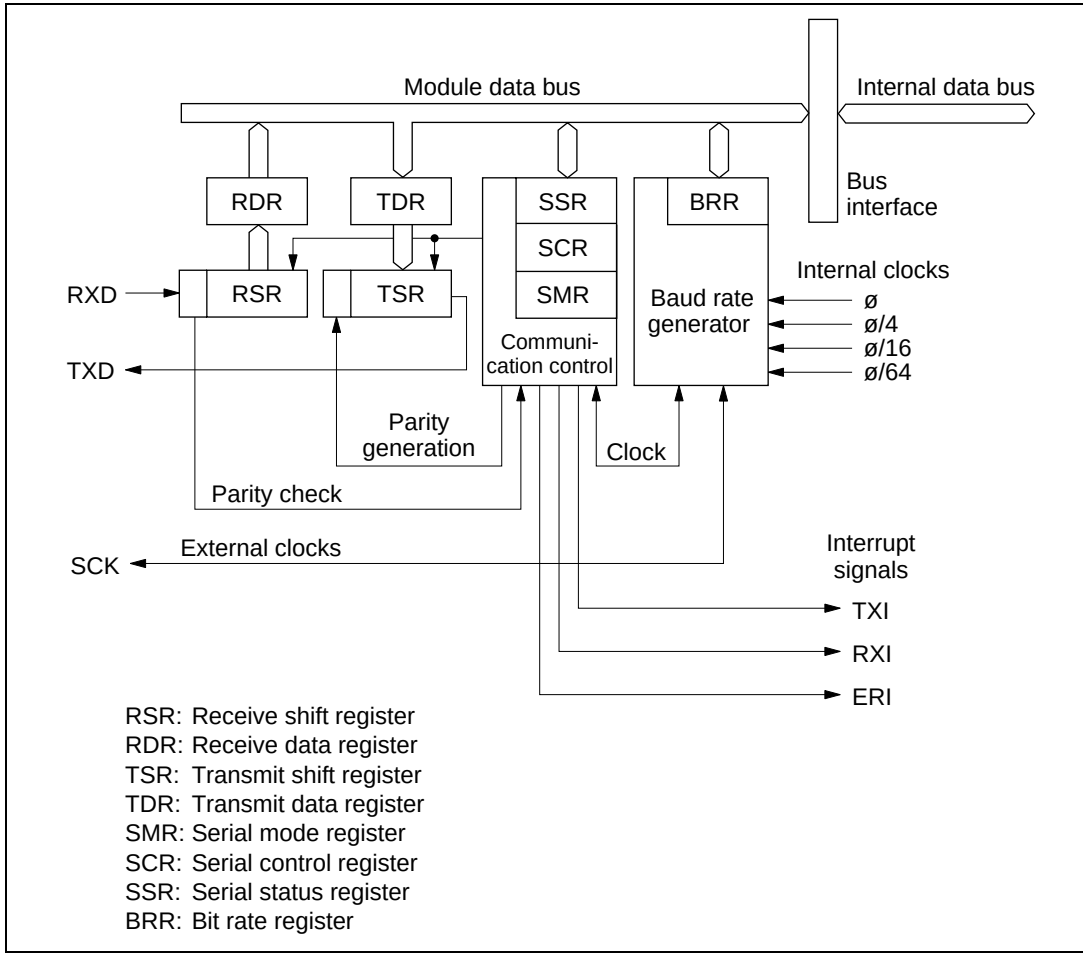


Figure 12.1 Block Diagram of Serial Communication Interface

12.1.3 Input and Output Pins

Name	Abbreviation	I/O	Function
Serial clock	SCK	Input/output	Serial clock input and output
Receive data	RXD	Input	Receive data input
Transmit data	TXD	Output	Transmit data output

12.1.4 Register Configuration

Name	Abbreviation	R/W	Initial Value	Address
Receive shift register	RSR	—	—	—
Receive data register	RDR	R	H'00	H'FE9D
Transmit shift register	TSR	—	—	—
Transmit data register	TDR	R/W	H'FF	H'FE9B
Serial mode register	SMR	R/W	H'04	H'FE98
Serial control register	SCR	R/W	H'0C	H'FE9A
Serial status register	SSR	R/(W)*	H'87	H'FE9C
Bit rate register	BRR	R/W	H'FF	H'FE99

Note: * By software, only a 0 can be written to clear the status flag bits.

12.2 Register Descriptions

12.2.1 Receive Shift Register (RSR)

Bit	7	6	5	4	3	2	1	0
Read/Write	—	—	—	—	—	—	—	—

RSR receives the incoming data bits. When one data character has been received, it is transferred to the receive data register (RDR).

The CPU cannot read from or write to RSR directly.

12.2.2 Receive Data Register (RDR)—H'FE9D

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R

RDR stores the received data. As each character is received, it is transferred from RSR to RDR, enabling RSR to receive the next character. This double-buffering allows the SCI to receive data continuously.

The CPU can only read from RDR. RDR is initialized to H'00 at reset and in the standby modes.

12.2.3 Transmit Shift Register (TSR)

Bit	7	6	5	4	3	2	1	0
Read/Write	—	—	—	—	—	—	—	—

TSR holds the character currently being transmitted. When transmission of this character is completed, the next character is moved from the transmit data register (TDR) to TSR and transmission of that character begins. If TDR does not contain valid data, the SCI stops transmitting.

The CPU cannot read from or write to TSR directly.

12.2.4 Transmit Data Register (TDR)—H'FE9B

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TDR is an 8-bit read/write register that holds the next character to be transmitted. When TSR becomes empty, the character written in TDR is transferred to TSR.

Continuous data transmission is possible by writing the next byte in TDR while the current byte is being transmitted from TSR.

TDR is initialized to H'FF at reset and in the standby modes.

12.2.5 Serial Mode Register (SMR)—H'FE98

Bit	7	6	5	4	3	2	1	0
	C/$\overline{A}$	CHR	PE	O/$\overline{E}$	STOP	—	CKS1	CKS0
Initial value	0	0	0	0	0	1	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	—	R/W	R/W

The SMR is an 8-bit read/write register that controls the communication format and selects the clock rate for the internal clock source. It is initialized to H'04 at reset and in the standby modes.

Bit 7—Communication Mode (C/$\overline{A}$): This bit selects the asynchronous or synchronous communication mode.

Bit 7 C/$\overline{A}$	Description
0	Asynchronous communication. (Also, the initial value.)
1	Communication is synchronized with the serial clock.

Bit 6—Character Length (CHR): This bit selects the character length in asynchronous mode. In synchronous mode this bit is ignored.

Bit 6 CHR	Description
0	8 bits per character. (Also, the initial value.)
1	7 bits per character.

Bit 5—Parity Enable (PE): This bit selects whether or not to add a parity bit in asynchronous mode. In synchronous mode this bit is ignored.

Bit 5 PE	Description
0	Transmit: No parity bit is added. (Also, the initial value.) Receive: Parity is not checked.
1	Transmit: A parity bit is added. Receive: Parity is checked.

Bit 4—Parity Mode (O/$\overline{E}$): In asynchronous mode when parity is enabled (PE = 1), this bit selects even or odd parity.

Bit 4 O/$\overline{E}$	Description
0	Even parity. (Also, the initial value.)
1	Odd parity.

Even parity meaning that a parity bit is added to the data bits for each character to make the total number of 1s even, and odd parity meaning that the total number of 1s is made odd.

This bit is ignored when PE = 0 and in the synchronous mode.

Bit 3—Stop Bit Length (STOP): This bit selects the number of stop bits. In synchronous mode this bit is ignored.

Bit 3 STOP	Description
0	1 stop bit. (Also, the initial value.)
1	2 stop bits.

Bit 2—Reserved: This bit cannot be modified and is always read as 1.

Bits 1 and 0—Clock Select 1 and 0 (CKS1 and CKS0): These bits select the internal clock source when the baud rate generator is clocked from within the H8/570 MCU.

Bit 1 CKS1	Bit 0 CKS0	Description
0	0	ø clock (Also, the initial value.)
	1	ø/4 clock
1	0	ø/16 clock
	1	ø/64 clock

12.2.6 Serial Control Register (SCR)—H'FE9A

Bit	7	6	5	4	3	2	1	0
	TIE	RIE	TE	RE	—	—	CKE1	CKE0
Initial value	0	0	0	0	1	1	0	0
Read/Write	R/W	R/W	R/W	R/W	—	—	R/W	R/W

SCR is an 8-bit read/write register that enables or disables various SCI functions. It is initialized to H'0C at reset and in the standby modes.

Bit 7—Transmit Interrupt Enable (TIE): This bit enables or disables the transmit-end interrupt (TXI) requested when the transmit data register empty (TDRE) bit in the serial status register (SSR) is set to 1.

Bit 7 TIE	Description
0	The transmit-end interrupt request (TXI) is disabled. (Also, the initial value.)
1	The transmit-end interrupt request (TXI) is enabled.

Bit 6—Receive Interrupt Enable (RIE): This bit enables or disables the receive-end interrupt (RXI) requested when the receive data register full (RDRF) bit in the serial status register (SSR) is set to 1. It also enables or disables the receive-error interrupt (ERI) request.

Bit 6 RIE	Description
0	The receive-end interrupt (RXI) and receive-error interrupt (ERI) requests are disabled. (Also, the initial value.)
1	The receive-end interrupt (RXI) and receive-error interrupt (ERI) requests are enabled.

Bit 5—Transmit Enable (TE): This bit enables or disables the transmit function. When the transmit function is enabled, the TXD pin is automatically used for output. When the transmit function is disabled, the TXD pin can be used as a general-purpose I/O port.

Bit 5 TE	Description
0	The transmit function is disabled and the TXD pin can be used as an I/O port. (Also, the initial value.)
1	The transmit function is enabled and the TXD pin is used for output.

Bit 4—Receive Enable (RE): This bit enables or disables the receive function. When the receive function is enabled, the RXD pin is automatically used for input. When the receive function is disabled, the RXD pin is available as a general-purpose I/O port.

Bit 4 RE	Description
0	The receive function is disabled and the RXD pin can be used as an I/O port. (Also, the initial value.)
1	The receive function is enabled and the RXD pin is used for input.

Bits 3 and 2—Reserved: These bits cannot be modified and are always read as 1.

Bit 1—Clock Enable 1 (CKE1): This bit selects the internal or external clock source for the baud rate generator. When external clock source is selected, the SCK pin automatically becomes an input pin for the external clock signal.

Bit 1 CKE1	Description
0	Internal clock source is selected. (Also, the initial value.)
1	External clock source is selected and the SCK pin is used for input.

Bit 0—Clock Enable 0 (CKE0): When an internal clock source is used in synchronous mode, this bit enables or disables serial clock output at the SCK pin.

This bit is ignored when external clock is selected, or when the asynchronous mode is selected.

For further information on the communication format and clock source selection, see tables 12.3 and 12.4 in section 12.3, Operation.

Bit 0 CKE0	Description
0	The SCK pin is not used by the SCI and is available as an I/O port. (Also, the initial value.)
1	The SCK pin is used for serial clock output.

12.2.7 Serial Status Register (SSR)—H'FE9C

Bit	7	6	5	4	3	2	1	0
	TDRE	RDRF	ORER	FER	PER	—	—	—
Initial value	1	0	0	0	0	1	1	1
Read/Write	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	—	—	—

Note: * By software, only a 0 can be written to clear the flags.

SSR is an 8-bit register that indicates transmit and receive status. It is initialized to H'87 at reset and in the standby modes.

Bit 7—Transmit Data Register Empty (TDRE): This bit indicates when the TDR contents have been transferred to TSR and the next character can safely be written into TDR.

Bit 7 TDRE	Description
0	This bit is cleared to 0 when: • The CPU reads the TDRE bit, then writes a 0 to this bit • The data transfer controller (DTC) writes data to TDR
1	This bit is set to 1 (the initial value) at the following times: • The MCU is reset or enters a standby mode • When the TDR contents are transferred to TSR • When TDRE = 0 and the TE bit is cleared to 0

Bit 6—Receive Data Register Full (RDRF): This bit indicates when one character has been received and transferred to RDR.

Bit 6 RDRF	Description
0	This bit is cleared to 0 (the initial value) when: • The CPU reads the RDRF bit, then writes a 0 to this bit • The data transfer controller (DTC) reads RDR • The MCU is reset or enters a standby mode
1	This bit is set to 1 when one character is received without error and transferred from RSR to RDR.

Bit 5—Overrun Error (ORER): This bit indicates an overrun error during reception.

Bit 5 ORER	Description
0	This bit is cleared to 0 (the initial value) when: • The CPU reads the ORER bit, then writes a 0 to this bit • The MCU is reset or enters a standby mode
1	This bit is set to 1 if the reception of the next character ends while the receive data register is still full (RDRF = 1).

Bit 4—Framing Error (FER): This bit indicates a framing error during data reception in the synchronous mode. It is not used in the asynchronous mode.

Bit 4 FER	Description
0	This bit is cleared to 0 (the initial value) when: • The CPU reads the FER bit, then writes a 0 to this bit • The MCU is reset or enters a standby mode
1	This bit is set to 1 if a framing error occurs (stop bit = 0).

Bit 3—Parity Error (PER): This bit indicates a parity error during data reception in the asynchronous mode when a communication format with parity bits is used.

This bit is not used in the synchronous mode, or when a communication format without parity bits is used.

Bit 3 PER	Description
0	This bit is cleared to 0 (the initial value) when: • The CPU reads the PER bit, then writes a 0 to this bit • The MCU is reset or enters a standby mode
1	This bit is set to 1 when a parity error occurs (the parity of the received data does not match the parity selected by the bit in SMR).

Bits 2 to 0—Reserved: These bits cannot be modified and are always read as 1.

12.2.8 Bit Rate Register (BRR)—H'FE99

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BRR is an 8-bit register that by combing it with the CKS1 and CKS0 bits in SMR, determines the bit rate output by the baud rate generator.

BRR is initialized to H'FF (slowest rate) at reset and in the standby modes.

Refer to tables 12.1 and 12.2 for examples of BRR (N) and CKS (n) settings for commonly used bit rates.

Table 12.1 Examples of BRR Settings in Asynchronous Mode

	XTAL Frequency (MHz)											
	2			2.4576			4			4.194304		
Bit Rate	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	1	174	–0.26	2	52	+0.50	2	64	+0.70	2	70	+0.03
110	1	70	+0.03	1	86	+0.31	1	141	+0.03	1	148	–0.04
150	0	207	+0.16	0	255	0	1	103	+0.16	1	108	+0.21
300	0	103	+0.16	0	127	0	0	207	+0.16	0	217	+0.21
600	0	51	+0.16	0	63	0	0	103	+0.16	0	108	+0.21
1200	0	25	+0.16	0	31	0	0	51	+0.16	0	54	–0.70
2400	0	12	+0.16	0	15	0	0	25	+0.16	0	26	+1.14
4800	—	—	—	0	7	0	0	12	+0.16	0	13	–2.48
9600	—	—	—	0	3	0	—	—	—	—	—	—
19200	—	—	—	0	1	0	—	—	—	—	—	—
31250	—	—	—	—	—	—	0	1	0	—	—	—
38400	—	—	—	0	0	0	—	—	—	—	—	—

	XTAL Frequency (MHz)											
	4.9152			6			7.3728			8		
Bit Rate	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	1	174	–0.26	2	52	+0.50	2	64	+0.70	2	70	+0.03
150	1	127	0	1	155	+0.16	1	191	0	1	207	+0.16
300	0	255	0	1	77	+0.16	1	95	0	1	103	+0.16
600	0	127	0	0	155	+0.16	0	191	0	0	207	+0.16
1200	0	63	0	0	77	+0.16	0	95	0	0	103	+0.16
2400	0	31	0	0	38	+0.16	0	47	0	0	51	+0.16
4800	0	15	0	0	19	–2.34	0	23	0	0	25	+0.16
9600	0	7	0	—	—	—	0	11	0	0	12	+0.16
19200	0	3	0	—	—	—	0	5	0	—	—	—
31250	—	—	—	0	2	0	—	—	—	0	3	0
38400	0	1	0	—	—	—	0	2	0	—	—	—

	XTAL Frequency (MHz)											
	9.8304			10			12			12.288		
Bit Rate	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	86	+0.31	2	88	–0.25	2	106	–0.44	2	108	+0.08
150	1	255	0	2	64	+0.16	2	77	0	2	79	0
300	1	127	0	1	129	+0.16	1	155	0	1	159	0
600	0	255	0	1	64	+0.16	1	77	0	1	79	0
1200	0	127	0	0	129	+0.16	0	155	+0.16	0	159	0
2400	0	63	0	0	64	+0.16	0	77	+0.16	0	79	0
4800	0	31	0	0	32	–1.36	0	38	+0.16	0	39	0
9600	0	15	0	0	15	+1.73	0	19	–2.34	0	19	0
19200	0	7	0	0	7	+1.73	—	—	—	0	4	0
31250	0	4	–1.70	0	4	0	0	5	0	0	5	+2.40
38400	0	3	0	0	3	+1.73	—	—	—	—	—	—

	XTAL Frequency (MHz)											
	14.7456			16			19.6608			20		
Bit Rate	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	130	–0.07	2	141	+0.03	2	174	–0.26	3	43	+0.88
150	2	95	0	2	103	+0.16	2	127	0	2	129	+0.16
300	1	191	0	1	207	+0.16	1	255	0	2	64	+0.16
600	1	95	0	1	103	+0.16	1	127	0	1	129	+0.16
1200	0	191	0	0	207	+0.16	0	255	0	1	64	+0.16
2400	0	95	0	0	103	+0.16	0	127	0	0	129	+0.16
4800	0	47	0	0	51	+0.16	0	63	0	0	64	+0.16
9600	0	23	0	0	25	+0.16	0	31	0	0	32	–1.36
19200	0	11	0	0	12	+0.16	0	15	0	0	15	+1.73
31250	—	—	—	0	7	0	0	9	–1.70	0	9	0
38400	0	5	0	—	—	—	0	7	0	0	7	+1.73

$$B = OSC \times 10^6/[64 \times 2^{2n} \times (N + 1)]$$

B: Bit rate
N: BRR value ($0 \leq N \leq 255$)
OSC: Crystal oscillator frequency in MHz
n: Internal clock source (0, 1, 2, or 3)

The specifications of n are:

n	CKS1	CKS0	Clock
0	0	0	ø
1	0	1	ø/4
2	1	0	ø/16
3	1	1	ø/64

Table 12.2 Examples of BRR Settings in Synchronous Mode

	XTAL Frequency (MHz)											
	2		4		8		10		16		20	
Bit Rate	**n**	**N**	**n**	**N**	**n**	**N**	**n**	**N**	**n**	**N**	**n**	**N**
100	—	—	—	—	—	—	—	—	—	—	—	—
250	1	249	2	124	2	249	—	—	3	124	—	—
500	1	124	1	249	2	124	—	—	2	249	—	—
1 k	0	249	1	124	1	249	—	—	2	124	—	—
2.5 k	0	99	0	199	1	99	1	124	1	199	1	249
5 k	0	49	0	99	0	199	0	249	1	99	1	124
10 k	0	24	0	49	0	99	0	124	0	199	0	249
25 k	0	9	0	19	0	39	0	49	0	79	0	99
50 k	0	4	0	9	0	19	0	24	0	39	0	49
100 k	—	—	0	4	0	9	—	—	0	19	0	24
250 k	0	0*	0	1	0	3	0	4	0	7	0	9
500 k			0	0*	0	1	—	—	0	3	0	4
1 M					0	0*	—	—	0	1	—	—
2.5 M											0	0*

Notes: Blank: No setting is available.
—: A setting is available, but the bit rate is inaccurate.
*: Data cannot be transferred continuously.

$$B = OSC/[8 \times 2^{2n} \times (N + 1)]$$

B: Bit rate
N: BRR value ($0 \leq N \leq 255$)
OSC: Crystal oscillator frequency in MHz
n: Internal clock source (0, 1, 2, or 3)

The specifications of n are:

n	CKS1	CKS0	Clock
0	0	0	ø
1	0	1	ø/4
2	1	0	ø/16
3	1	1	ø/64

12.3 Operation

12.3.1 Overview

The SCI supports serial data transfer in both asynchronous and synchronous modes.

The communication format depends on the settings in SMR (table 12.3). The clock source and usage of the SCK pin depend on the settings in SMR and SCR (table 12.4).

Table 12.3 Communication Formats Used by SCI

	SMR						
C/$\overline{A}$	**CHR**	**PE**	**STOP**	**Mode**	**Format**	**Parity**	**Stop Bit Length**
0	0	0	0	Asynchronous	8-bit data	None	1
			1				2
		1	0			Yes	1
			1				2
	1	0	0		7-bit data	None	1
			1				2
		1	0			Yes	1
			1				2
1	—	—	—	Synchronous	8-bit data	—	—

Table 12.4 SCI Clock Source Selection

SMR	**SCR**			
C/A	**CKE1**	**CKE0**	**Clock Source**	**SCK Pin**
0 (Async mode)	0	0	Internal	I/O port*
		1		Clock output at same frequency as baud rate
	1	0	External	Clock input at 16 times the baud rate frequency
		1		
1 (Sync mode)	0	0	Internal	Serial clock output
		1		
	1	0	External	External Serial clock input
		1		

Note: * The SCK pin cannot be used by the SCI.

12.3.2 Asynchronous Mode

In asynchronous mode, each character is individually synchronized by framing it with a start bit and stop bit(s) (figure 12.2).

Full-duplex data transfer is possible since the SCI has independent transmit and receive sections. Double buffering in both sections enables the SCI to be programmed for continuous data transfer.

The communication channel is normally held in the “mark” or high state. Character transmission or reception starts with a transition to the “space” or low state.

The first bit transmitted or received is the start bit (low). It is followed by the data bits, which are led by the least significant bit. The data bits are then followed by the parity bit, if present, then the stop bit(s) (high) confirming the end of the frame.

In receiving, the SCI synchronizes on the falling edge of the start bit, and samples each bit at its center (i.e., at the 8th clock cycle for a bit period of 16 clock cycles).

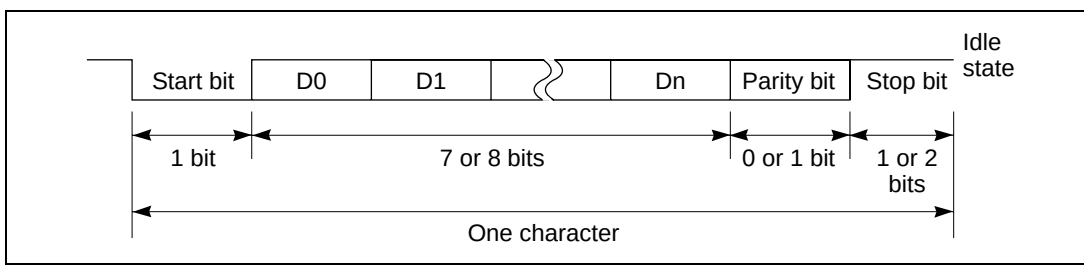


Figure 12.2 Data Format in Asynchronous Mode

Data Format: Eight data formats (table 12.5) can be sent and received in asynchronous mode as selected by bits in SMR.

Table 12.5 Data Formats in Asynchronous Mode

SMR Bits			
CHR	**PE**	**STOP**	**Data Format**
0	0	0	Start | 8-bit data | Stop
0	0	1	Start | 8-bit data | Stop | Stop
0	1	0	Start | 8-bit data | P | Stop
0	1	1	Start | 8-bit data | P | Stop | Stop
1	0	0	Start | 7-bit data | Stop
1	0	1	Start | 7-bit data | Stop | Stop
1	1	0	Start | 7-bit data | P | Stop
1	1	1	Start | 7-bit data | P | Stop | Stop

Notes: Start: Start bit
Stop: Stop bit
P: Parity bit

Clock: In the asynchronous mode it is possible to select either an internal clock created by the baud rate generator, or an external clock input at the SCK pin. Refer to table 12.4.

If an external clock is input at the SCK pin, its frequency should be 16 times the desired baud rate.

If the internal clock provided by the baud rate generator is selected and the SCK pin is used for clock output, the output clock frequency will equal the baud rate, and the clock pulse will rise at the center of the transmit data bits (figure 12.3).

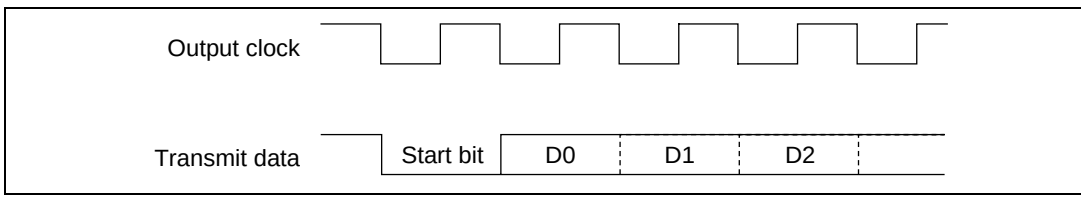


Figure 12.3 Phase Relationship Between Clock Output and Transmit Data

Data Transmission and Reception:

• SCI initialization

Before data can be transmitted or received, the SCI must be initialized by software. To initialize the SCI, the program must clear the TE and RE bits to 0, then execute the following procedure.

1. Set the desired communication format in SMR.
2. Write the value corresponding to the desired bit rate in BRR. (This step is not necessary if an external clock is used.)
3. Select the clock and enable the desired interrupts in SCR.
4. Set the TE and/or RE bit in SCR to 1.

The TE and RE bits must both be cleared to 0 whenever the operating mode or data format is changed.

After changing the operating mode or data format and before setting the TE and RE bits to 1, the program must be delayed for at least the transfer time of 1 bit at the selected baud rate, in order to ensure that the SCI is initialized. If an external clock is used, the clock must not be stopped.

When clearing the TDRE bit during data transmission to assure the transfer of correct data, do not clear the TDRE bit until after writing data into TDR. Similarly, for receiving data, do not clear the RDRF bit until after reading data from RDR.

• Data transmission

The procedure for transmitting data is as follows.

1. Set up the desired transmitting conditions in SMR, SCR, and BRR.
2. Set the TE bit in SCR to 1. The TXD pin will automatically become an output pin and one frame* of all 1s will be transmitted, after which the SCI is ready to transmit data.
3. Check that the TDRE bit is set at 1, then write the first byte of transmit data into TDR. Next clear the TDRE bit to 0.
4. The first byte of transmit data is transferred from TDR to TSR and sent in the designated format as follows:

   Start bit: One 0 bit.

   Transmit data: Seven or eight bits starting from bit 0.

   Parity bit: Odd parity, even parity, or no parity bit.

   Stop bit: One or two consecutive 1 bits.
5. The transfer of the transmit data from TDR to TSR makes TDR empty, so the TDRE bit is set to 1. If the TIE bit is set at 1, a transmit-end interrupt (TXI) is requested. When the transmit function is enabled but TDR is empty (TDRE = 1), the output at the TXD pin is held at 1 until the TDRE bit is cleared to 0.

Note: * A frame is the data for one character, including the start bit and stop bit(s).

• Data reception

The procedure for receiving data is as follows.

1. Set up the desired receiving conditions in SMR, SCR, and BRR.
2. Set the RE bit in SCR to 1. The RXD pin will automatically become an input pin and the SCI will be ready to receive data.
3. The SCI synchronizes with the incoming data by detecting the start bit, and places the received bits in RSR. At the end of the data, the SCI checks if the stop bit is 1. If the stop bit length is 2 bits, the SCI checks if both bits are 1.
4. When a complete frame has been received, the SCI transfers the received data to RDR so that it can be read. If the character length is 7 bits, the most significant bit of RDR is cleared to 0. At the same time, the SCI sets the RDRF bit in SSR to 1. If the RIE bit is set to 1, a receive-end interrupt (RXI) is requested.
5. The RDRF bit is cleared to 0 either when the CPU reads SSR and a 0 is written as the RDRF bit, or when RDR is read by the data transfer controller (DTC). RDR is then ready to receive the next character from RSR. When a frame is not received correctly, one of three receive errors occur (table 12.6).

If a receive error occurs, the RDRF bit in SSR is not set to 1. Instead, the corresponding error flag is set to 1. If the RIE bit in SCR is set to 1, a receive-error interrupt (ERI) is requested.

When a framing or parity error occurs, the RSR contents are transferred to RDR. If an overrun error occurs, however, the RSR contents are not transferred to RDR.

If multiple receive errors occur simultaneously, all the corresponding error flags are set to 1.

To clear a receive-error flag (ORER, FER, or PER), the program must read SSR, then write a 0 into the flag bit.

Table 12.6 Receive Errors

Name	Abbreviation	Description
Overrun error	ORER	Reception of the next frame ends while the RDRF bit is still set at 1. The RSR contents are not transferred to RDR.
Framing error	FER	A stop bit is 0. The RSR contents are transferred to RDR.
Parity error	PER	The parity of a frame does not match the value selected by the bit in SMR. The RSR contents are transferred to RDR.

12.3.3 Synchronous Mode

The synchronous mode is suitable for high-speed, continuous data transfers. Each bit of the data is synchronized with a serial clock pulse.

Continuous data transfer is enabled by the double buffering employed in both the transmit and receive sections of the SCI. Full duplex communication is possible due to independent transmit and receive sections.

Data Format: The communication format (figure 12.4) used in the synchronous mode consists of a data length of 8 bits for both the transmit and receive directions. The least significant bit is sent and received first. Each bit of the transmit data is output from the falling edge of the serial clock pulse to the next falling edge. Received bits are latched on the rising edge of the serial clock pulse.

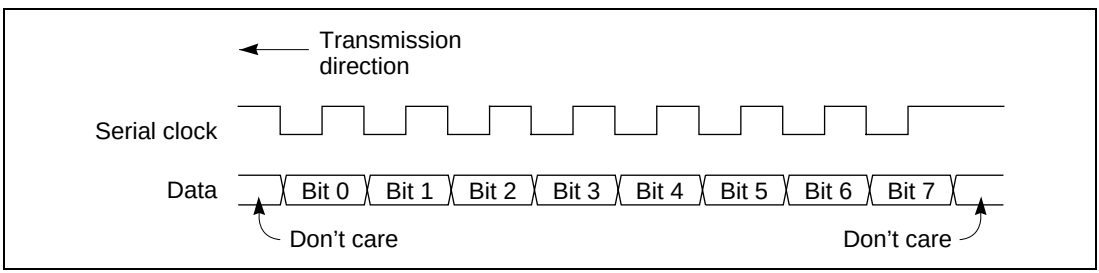


Figure 12.4 Data Format in Synchronous Mode

Clock: Either the internal serial clock created by the baud rate generator or an external clock input at the SCK pin can be selected in the synchronous mode. See table 12.4 for details.

Data Transmission and Reception:

• SCI initialization

Before data can be transmitted or received, the SCI must be initialized by software. To initialize the SCI, the program must clear the TE and RE bits to 0 to disable both the transmit and receive functions, then the following procedure must be executed.

1. Write the value corresponding to the desired bit rate into BRR. (This step is not necessary if an external clock is used.)
2. Select the clock in SCR.
3. Select the synchronous mode in SMR.
4. Set the TE and/or RE bit to 1, and enable the desired interrupts in SCR.

The TE and RE bits must both be cleared to 0 whenever the operating mode or data format is to be changed. After changing the operating mode or data format and before setting the TE and RE bits back to 1, the program must wait for at least 1 bit transfer time at the selected communication speed to ensure that the SCI is initialized.

When clearing the TDRE bit during data transmission to assure correct data transfer, do not clear the TDRE bit until after writing data into TDR. Similarly, for receiving data, do not clear the RDRF bit until after reading data from RDR.

• Data transmission

The procedure for transmitting data is as follows.

1. Set up the desired transmitting conditions in SMR, BRR, and SCR.
2. Set the TE bit in SCR to 1. The TXD pin will automatically become an output pin, after which the SCI will be ready to transmit data.
3. Make sure that the TDRE bit is set to 1, then write the first byte of transmit data into TDR. Next, clear the TDRE bit to 0.
4. The first byte of transmit data is transferred from TDR to TSR and sent as each bit is synchronized with a clock pulse. Bit 0 is sent first. Since the transfer of the transmit data from TDR to TSR empties TDR, the TDRE bit is set to 1. If the TIE bit is set at 1, a transmit-end interrupt (TXI) is requested.

TDR and TSR function as a double buffer. Continuous data transmission can be achieved by writing the next transmit data into TDR and clearing the TDRE bit to 0 while the SCI is transmitting the current data from TSR.

If an internal clock source is selected, after transferring the transmit data from TDR to TSR and while transmitting the data from TSR, the SCI also outputs a serial clock signal at the SCK pin. When all data bits in TSR have been transmitted and if TDR is empty (TDRE = 1), serial clock output is suspended until the next data byte is written into TDR and the TDRE bit is cleared to 0. During this interval the TXD pin is held at the value of the last bit transmitted.

If an external clock source is selected, data transmission is synchronized with the clock signal input at the SCK pin. When all data bits in TSR have been transmitted and if TDR is empty (TDRE = 1) but external clock pulses continue to arrive, the TXD pin outputs a string of bits equal to the last bit transmitted.

• Data reception

The procedure for receiving data is as follows.

1. Set up the desired receiving conditions in SMR, BRR, and SCR.
2. Set the RE bit in SCR to 1. The RXD pin will automatically become an input pin and the SCI will be ready to receive data.
3. Incoming data bits are latched in RSR on eight clock pulses. When 8 bits of data have been received, the SCI sets the RDRF bit in SSR to 1. If the RIE bit is set at 1, a receive-end interrupt (RXI) is requested.

4. The SCI transfers the received data byte to RDR so that it can be read. The RDRF bit is cleared either when the program reads the RDRF bit in SSR, then writes a 0 in the RDRF bit, or when the data transfer controller (DTC) reads RDR.

RDR and RSR function as a double buffer. Data can be received continuously by reading each byte of data from RDR and clearing the RDRF bit to 0 before the last bit of the next byte is received.

In general, an external clock source should be used for receiving data.

If an internal clock source is selected, the SCI starts receiving data as soon as the RE bit is set to 1. The serial clock is also output at the SCK pin. The SCI continues receiving data until the RE bit is cleared to 0.

If the last bit of the next data byte is received while the RDRF bit is still set at 1, an overrun error occurs and the ORER bit is set to 1. If the RIE bit is set at 1, a receive-error interrupt (ERI) is requested. The data received in RSR is not transferred to RDR when an overrun error occurs.

After the occurrence of an overrun error, the reception of the next data is enabled when the ORER bit is cleared to 0.

• Simultaneous transmit and receive

The procedure for transmitting and receiving simultaneously is as follows:

1. Set up the desired communication conditions in SMR, BRR, and SCR.
2. Set the TE and RE bits in SCR to 1. The TXD and RXD pins will automatically become output and input pins, respectively, and the SCI will be ready to transmit and receive data.
3. Data transmitting and receiving start when the TDRE bit in SSR is cleared to 0.
4. Data is sent and received in synchronization with eight clock pulses.
5. First, the transmit data is transferred from TDR to TSR. This empties TDR so the TDRE bit is then set to 1. If the TIE bit is set at 1, a transmit-end interrupt (TXI) is requested. If continuous data transmission is desired, the CPU must read the TDRE bit in SSR, write the next transmit data into TDR, then clear the TDRE bit to 0.

   Alternatively, the DTC can write the next transmit data into TDR, in which case the TDRE bit is cleared automatically. If the TDRE bit is not cleared to 0 by the time the SCI completes the transmission of the current byte from TSR, the TXD pin will continue to output the last bit in TSR.
6. In the receiving section, when 8 bits of data have been received they are transferred from RSR to RDR and the RDRF bit in SSR is set to 1. If the RIE bit is set at 1, a receive-end interrupt (RXI) is requested.

7. To clear the RDRF bit, the program must read the RDRF bit in SSR, read the data in RDR, then write a 0 into the RDRF bit. Alternatively, the DTC can read RDR, in which case the RDRF bit is cleared automatically. For continuous data reception, the RDRF bit must be cleared to 0 before the last bit of the next byte of data is received.

If the last bit of the next byte is received while the RDRF bit is still set at 1, an overrun error occurs. The error is handled with the same procedure as the data reception described previously. The overrun error does not affect the transmit section of the SCI, which continues to transmit normally.

12.4 CPU Interrupts and DTC Interrupts

The SCI can request three types of interrupts: transmit-end (TXI), receive-end (RXI), and receive-error (ERI). Interrupt requests are enabled or disabled by the TIE and RIE bits in SCR. Independent signals are sent to the interrupt controller for each of these interrupts. The transmit-end and receive-end interrupt request signals are obtained from the TDRE and RDRF flags. The receive-error interrupt request signal is the result of the logical-OR of three error flags: overrun error (ORER), framing error (FER), and parity error (PER). See table 12.7.

Table 12.7 SCI Interrupts

Interrupt	Description	DTC Service Available?	Priority
ERI	Receive-error interrupt, requested when ORER, FER, or PER is set.	No	High
RXI	Receive-end interrupt, requested when RDRF is set.	Yes	↑
TXI	Transmit-end interrupt, requested when TDRE is set.	Yes	Low

The TXI and RXI interrupts can be served by the data transfer controller (DTC) to perform data transfer. When the DTC serves one of these interrupts, it clears the TDRE or RDRF bit to 0 under the following conditions, which differ between the two bits.

When invoked by a TXI request and if the DTC writes to TDR, the DTC automatically clears the TDRE bit to 0. When invoked by an RXI request and if the DTC reads from RDR, the DTC automatically clears the RDRF bit to 0.

See section 6, Data Transfer Controller, for further information on the DTC.

12.5 Application Notes

Application programmers should note the following features of the SCI.

TDR Write: The TDRE bit in SSR is simply a flag that indicates that the TDR contents have been transferred to TSR. The TDR contents can be rewritten regardless of the TDRE value. If a new byte is written into TDR while the TDRE bit is 0 and before the old TDR contents have been moved into TSR, the old byte will be lost. Normally, the program should check if the TDRE bit is set at 1 before writing to TDR.

Multiple Receive Errors: When multiple receive errors occur (table 12.8), the values of flag bits in SSR change accordingly, and the contents of RSR are not always transferred to RDR.

Table 12.8 SSR Bit States and Data Transfer When Multiple Receive Errors Occur

	SSR Bits				
Receive Error	**RDRF**	**ORER**	**FER**	**PER**	**RSR to RDR*[2]**
Overrun error	1*[1]	1	0	0	No
Framing error	0	0	1	0	Yes
Parity error	0	0	0	1	Yes
Overrun + framing errors	1*[1]	1	1	0	No
Overrun + parity errors	1*[1]	1	0	1	No
Framing + parity errors	0	0	1	1	Yes
Overrun + framing + parity errors	1*[1]	1	1	1	No

Notes: 1. Set to 1 before the overrun error occurs.
2. Yes: The RSR contents are transferred to RDR.
No: The RSR contents are not transferred to RDR.

Line Break Detection: When the RXD pin receives a continuous stream of 0s in the asynchronous mode (line-break state), a framing error occurs because of a 0 stop bit detection by the SCI. The value H'00 is then transferred from RSR to RDR. The program can detect the line-break state as a framing error by the accompanied H'00 data in RDR.

The SCI continues to receive data, so if the FER bit is cleared to 0 another framing error will occur.

Sampling Timing and Receive Margin in Asynchronous Mode: The serial clock used by the SCI in asynchronous mode runs at 16 times the bit rate. The falling edge of the start bit is detected by sampling the D input on the falling edge of this clock. After the start bit is detected, each bit of receive data in the frame (including the start bit, parity bit, and stop bit(s)) is sampled on the rising edge of the serial clock pulse being at the center of the bit. See figure 12.5.

The receive margin can be calculated as in equation (1).

When the absolute frequency deviation of the clock signal is 0 and the clock duty factor is 0.5, data can be theoretically received with a distortion of up to the margin given by equation (2). However, since this is a theoretical limit, in practice, system designers should allow a margin of 20% to 30%.

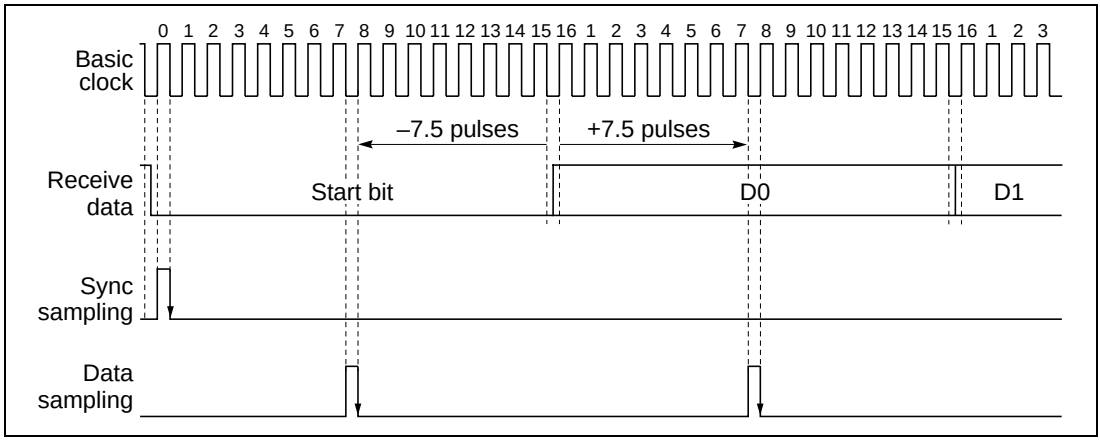

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 1 2 3
Basic clock
–7.5 pulses
+7.5 pulses
Receive data
Start bit
D0
D1
Sync sampling
Data sampling

Figure 12.5 Sampling Timing (Asynchronous Mode)

$$M = \{(0.5 - 1/2N) - (D - 0.5)/N - (L - 0.5)F\} \times 100\ [\%] \quad \text{..... (1)}$$

M: Receive margin
N: Ratio of basic clock to bit rate (16)
D: Duty factor of clock—ratio of high pulse width to low width (0.5 to 1.0)
L: Frame length (9 to 12)
F: Absolute clock frequency deviation

When D = 0.5 and F = 0,

$$M = (0.5 - 1/2 \times 16) \times 100\ [\%] = 46.875\% \quad \text{..... (2)}$$

Section 13 A/D Converter

13.1 Overview

The H8/570 MCU includes an analog-to-digital converter module which can be programmed for input of analog signals up to eight channels. A/D conversion is performed by the successive approximations method with 10-bit resolution.

13.1.1 Features

- Eight analog input channels
- Sample-and-hold circuit
- 10-bit resolution
- Rapid conversion time: 13.4 µs per channel (ø = 10 MHz)
- Single and scan modes
  - Single mode: A/D conversion is performed once.
  - Scan mode: A/D conversion is performed in a repeated cycle on one to four channels.
- Four 16-bit data registers: Storage of A/D conversion results for all four channels.
- CPU interrupt (ADI) request at completion of each A/D conversion cycle: Also used by the data transfer controller (DTC) providing a convenient way to move results into memory.

13.1.2 Block Diagram

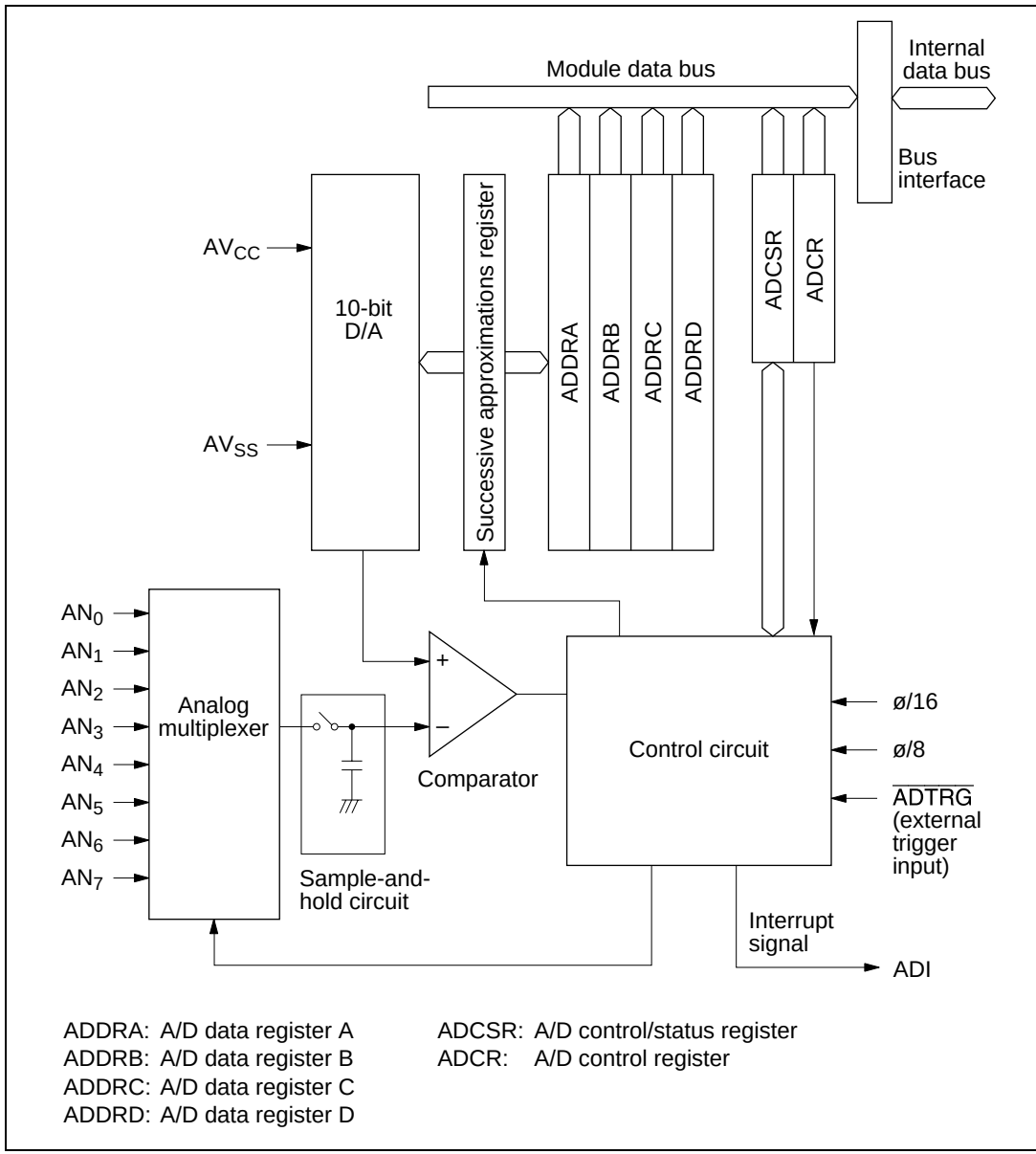


Figure 13.1 Block Diagram of A/D Converter

13.1.3 Input Pins

The eight analog input pins (table 13.1) are divided into two groups, consisting of analog inputs 0 to 3 (AN_0 to AN_3) and analog inputs 4 to 7 (AN_4 to AN_7), respectively.

Table 13.1 A/D Input Pins

Name	Abbreviation	I/O	Function
Analog supply voltage	AV_{CC}	Input	Power supply and reference voltage for the analog circuits
Analog ground	AV_{SS}	Input	Ground and reference voltage for the analog circuits
Analog input 0	AN_0	Input	Analog input pins, group 0
Analog input 1	AN_1	Input	
Analog input 2	AN_2	Input	
Analog input 3	AN_3	Input	
Analog input 4	AN_4	Input	Analog input pins, group 1
Analog input 5	AN_5	Input	
Analog input 6	AN_6	Input	
Analog input 7	AN_7	Input	
A/D external trigger input	$\overline{ADTRG}$	Input	External trigger input

Table 13.2 Configuration of A/D Registers

Name	Abbreviation	R/W	Initial Value	Address
A/D data register A (high)	ADDRA (H)	R	H'00	H'FE80
A/D data register A (low)	ADDRA (L)	R	H'00	H'FE81
A/D data register B (high)	ADDRB (H)	R	H'00	H'FE82
A/D data register B (low)	ADDRB (L)	R	H'00	H'FE83
A/D data register C (high)	ADDRC (H)	R	H'00	H'FE84
A/D data register C (low)	ADDRC (L)	R	H'00	H'FE85
A/D data register D (high)	ADDRD (H)	R	H'00	H'FE86
A/D data register D (low)	ADDRD (L)	R	H'00	H'FE87
A/D control/status register	ADCSR	R/(W)*	H'00	H'FE88
A/D control register	ADCR	R/W	H'7F	H'FE89

Note: * By software, only a 0 can be written to clear the status flag bits.

13.2 Register Descriptions

13.2.1 A/D Data Registers (ADDR)—H'FE80 to H'FE87

The four A/D data registers (ADDRA to ADDRD) are 16-bit read-only registers that store the results of the A/D conversion.

Bit	7	6	5	4	3	2	1	0
ADDRn H	AD_9	AD_8	AD_7	AD_6	AD_5	AD_4	AD_3	AD_2
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R
(n = A to D)								

Bit	7	6	5	4	3	2	1	0
ADDRn L	AD_1	AD_0	—	—	—	—	—	—
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R
(n = A to D)								

Each result consists of 10 bits. The first 8 bits are stored in the upper byte of the data register corresponding to the selected channel. The last two bits are stored in the lower data register byte. Each data register is assigned to two analog input channels (table 13.3).

The A/D data registers are always readable by the CPU. The upper byte can be read directly and the lower byte is read via a temporary register. See section 13.3, CPU Interface, for details.

The unused bits (bits 5 to 0) of the lower data register byte are always read as 0.

The A/D data registers are initialized to H'0000 at reset and in the standby modes.

Table 13.3 Assignment of Data Registers to Analog Input Channels

Analog Input Channel		
Group 0	**Group 1**	**A/D Data Register**
AN_0	AN_4	ADDRA
AN_1	AN_5	ADDRB
AN_2	AN_6	ADDRC
AN_3	AN_7	ADDRD

13.2.2 A/D Control/Status Register (ADCSR)—H'FE88

Bit	7	6	5	4	3	2	1	0
	ADF	ADIE	ADST	SCAN	CKS	CH2	CH1	CH0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/(W)*	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: * By software, only a 0 can be written into bit 7 to clear the flag.

The A/D control/status register (ADCSR) is an 8-bit read/write register that controls the operation of the A/D converter module.

ADCSR is initialized to H'00 at reset and in the standby modes.

Bit 7—A/D End Flag (ADF): This status flag indicates the end of one cycle of A/D conversion.

Bit 7 ADF	Description
0	This bit is cleared to 0 (the initial value) when: • The MCU is reset or enters a standby mode • The CPU reads the ADF bit, then writes a 0 into this bit • An A/D interrupt is served by the data transfer controller (DTC)
1	This bit is set to 1 at the following times: • Single mode: When one A/D conversion is completed. • Scan mode: When inputs on all selected channels have been converted.

Bit 6—A/D Interrupt Enable (ADIE): This bit selects whether or not to request an A/D interrupt (ADI) when A/D conversion is completed.

Bit 6 ADIE	Description
0	The A/D interrupt request (ADI) is disabled. (Also, the initial value.)
1	The A/D interrupt request (ADI) is enabled.

Bit 5—A/D Start (ADST): The A/D converter operates whenever this bit is set to 1. In the single mode, this bit is automatically cleared to 0 at the end of each A/D conversion.

Bit 5 ADST	Description
0	A/D conversion is halted. (Also, the initial value.)
1	• Single mode: One A/D conversion is performed. This bit is automatically cleared to 0 at the end of the conversion. • Scan mode: A/D conversion starts and continues cyclically on the selected channels until this bit is cleared to 0.

Bit 4—Scan Mode (SCAN): This bit selects either scan mode or single mode for operation. See section 13.4, Operation, for descriptions of these modes. The mode should be changed only when the ADST bit is cleared to 0.

Bit 4 SCAN	Description
0	Single mode. (Also, the initial value.)
1	Scan mode.

Bit 3—Clock Select (CKS): This bit controls the A/D conversion time.

The conversion time should be changed only when the ADST bit is cleared to 0.

Bit 3 CKS	Description
0	Conversion time = 266 states. (Also, the initial value.)
1	Conversion time = 134 states.

Bits 2 to 0—Channel Select 2 to 0 (CH2 to CH0): These bits and the SCAN bit combine to select one or more analog input channels.

The channel selection should be changed only when the ADST bit is cleared to 0.

Group Select	Channel Select		Selected Channels	
CH2	CH1	CH0	Single Mode	Scan Mode
0	0	0	AN_0	AN_0
		1	AN_1	AN_0 and AN_1
	1	0	AN_2	AN_0 to AN_2
		1	AN_3	AN_0 to AN_3
1	0	0	AN_4	AN_4
		1	AN_5	AN_4 and AN_5
	1	0	AN_6	AN_4 to AN_6
		1	AN_7	AN_4 to AN_7

13.2.3 A/D Control Register (ADCR)—H'FE89

Bit	7	6	5	4	3	2	1	0
	TRGE	—	—	—	—	—	—	—
Initial value	0	1	1	1	1	1	1	1
Read/Write	R/W	—	—	—	—	—	—	—

The A/D control register (ADCR) is an 8-bit read/write register that selects whether or not to start the A/D conversion when an external trigger is input.

ADCR is initialized to H'7F at reset and in the standby modes.

Bit 7—Trigger Enable (TRGE): This bit selects whether or not to start the A/D conversion when an external trigger is input.

Bit 7 TRGE	Description
0	When an external trigger is input, the A/D conversion does not start. (Also, the initial value.)
1	The A/D conversion starts at the falling edge of an input signal from the external trigger pin ($\overline{\text{ADTRG}}$).

Bit 6 to 0—Reserved: These bits cannot be modified and are always read as 1.

13.3 CPU Interface

The A/D data registers (ADDRA to ADDRD) are 16-bit registers. The upper byte of each register can be read directly, but the lower byte is accessed through an 8-bit temporary register (TEMP).

When the CPU or DTC reads the upper byte of an A/D data register while at the same time the upper byte is placed on the internal data bus, the lower byte is transferred to TEMP. When the lower byte is accessed, the value in TEMP is placed on the internal data bus. See figure 13.2.

A program that requires all 10 bits of an A/D result should perform word access, or should first read the upper byte, then the lower byte of the A/D data register. Either method would assure the CPU of obtaining consistent data. If the program reads the lower byte first, consistent data will not be guaranteed.

A program that requires only the accuracy of 8-bit A/D results should perform byte access to the upper byte of the A/D data register. The value in TEMP can be left unread.

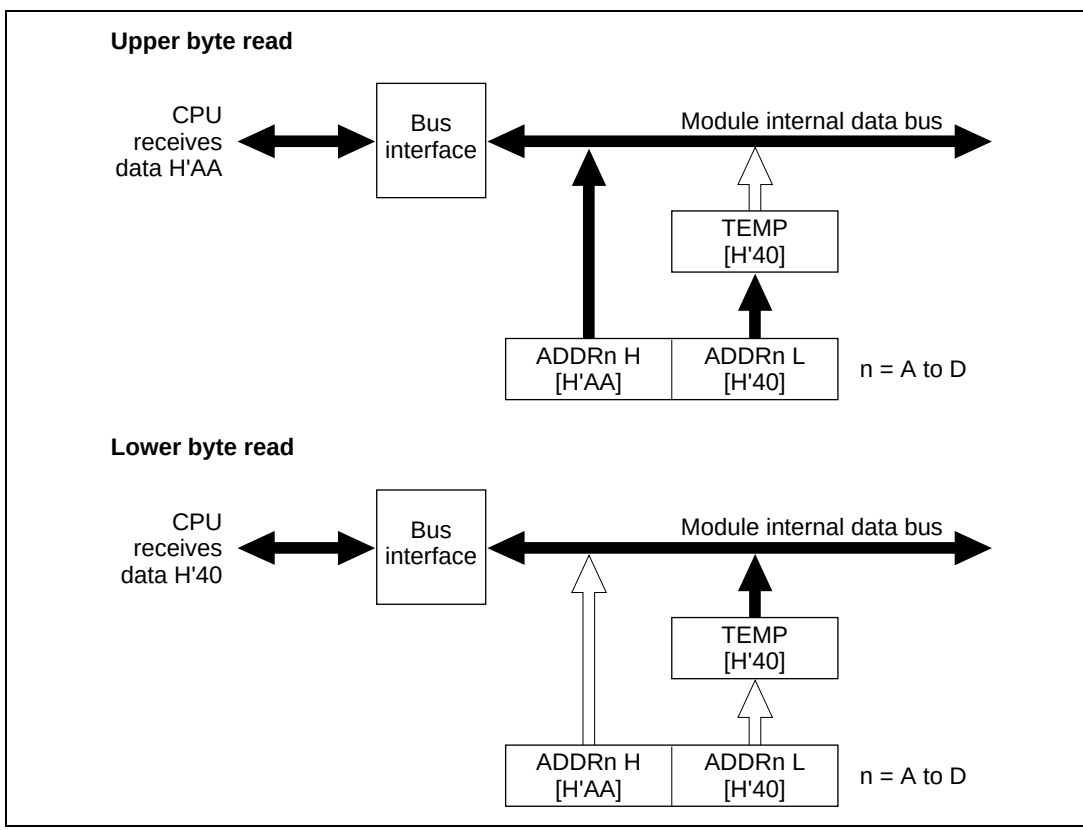


Figure 13.2 Read Access to A/D Data Register (When Register Contains H'AA40)

13.4 Operation

The A/D converter performs 10 successive approximations to obtain a result ranging from H'0000 (AV_{SS}) to H'FFC0 (AV_{CC}). Only the first 10 bits of the result are significant.

The A/D converter module can be programmed to operate in single mode or scan mode.

13.4.1 Single Mode

The single mode is suitable for obtaining a single data value from a single channel. A/D conversion starts when the ADST bit is set to 1 by software or an external trigger input. During the conversion process the ADST bit remains set at 1. When conversion is completed, the ADST bit is automatically cleared to 0.

When the conversion is completed, the ADF bit is set to 1. If the interrupt enable bit (ADIE) is also set to 1, an A/D conversion end interrupt (ADI) is requested so that the converted data can be processed by an interrupt-handling routine. Alternatively, the interrupt can be served by the data transfer controller (DTC).

When an A/D interrupt is served by the DTC, the DTC automatically clears the ADF bit to 0. When an A/D interrupt is served by the CPU, however, the ADF bit remains set until the CPU reads ADCSR, then writes a 0 into the ADF bit.

Before selecting single mode, clock, and analog input channel, the program should clear the ADST bit to 0 to ensure that the A/D converter has stopped. Changing the mode, clock, or channel selection while the A/D conversion is in progress can generate conversion errors.

The following is an example of the A/D conversion process in single mode when channel 1 (AN_1) is selected. See figure 13.3.

1. The program clears the ADST bit to 0, selects single mode (SCAN = 0) and channel 1 (CH2 to CH0 = 001), enables the A/D interrupt request (ADIE = 1), and sets the ADST bit to 1 to start the A/D conversion. (Selection of mode, clock, and channel, and the setting of the ADST bit can be done simultaneously.)

   Coding Example: (when using a slow clock, CKS = 0)

   ```
   BCLR  #5,@H'FE88
   MOV.B #H'61,@H'FE88
   ```

2. The A/D converter samples the AN_1 input and converts the voltage level to a digital value. At the end of the conversion process the A/D converter transfers the result to register ADDRB, sets the ADF bit to 1, clears the ADST bit to 0, and halts.
3. Since ADF = 1 and ADIE = 1, an A/D interrupt is requested.
4. The user-coded A/D interrupt-handling routine is started.

5. The interrupt-handling routine reads the ADCSR value, then writes a 0 into the ADF bit.
6. The interrupt-handling routine reads and processes the A/D conversion result.
7. The routine ends.

Steps (2) to (7) can now be repeated by setting the ADST bit to 1 again.

If the data transfer enable (DTE) bit is set to 1, the interrupt is served by the data transfer controller (DTC). Steps (4) to (7) then change as follows.

4'. The DTC is started.
5'. The DTC automatically clears the ADF bit to 0.
6'. The DTC transfers the A/D conversion result from ADDRB to a specified destination address.
7'. The DTC stops.

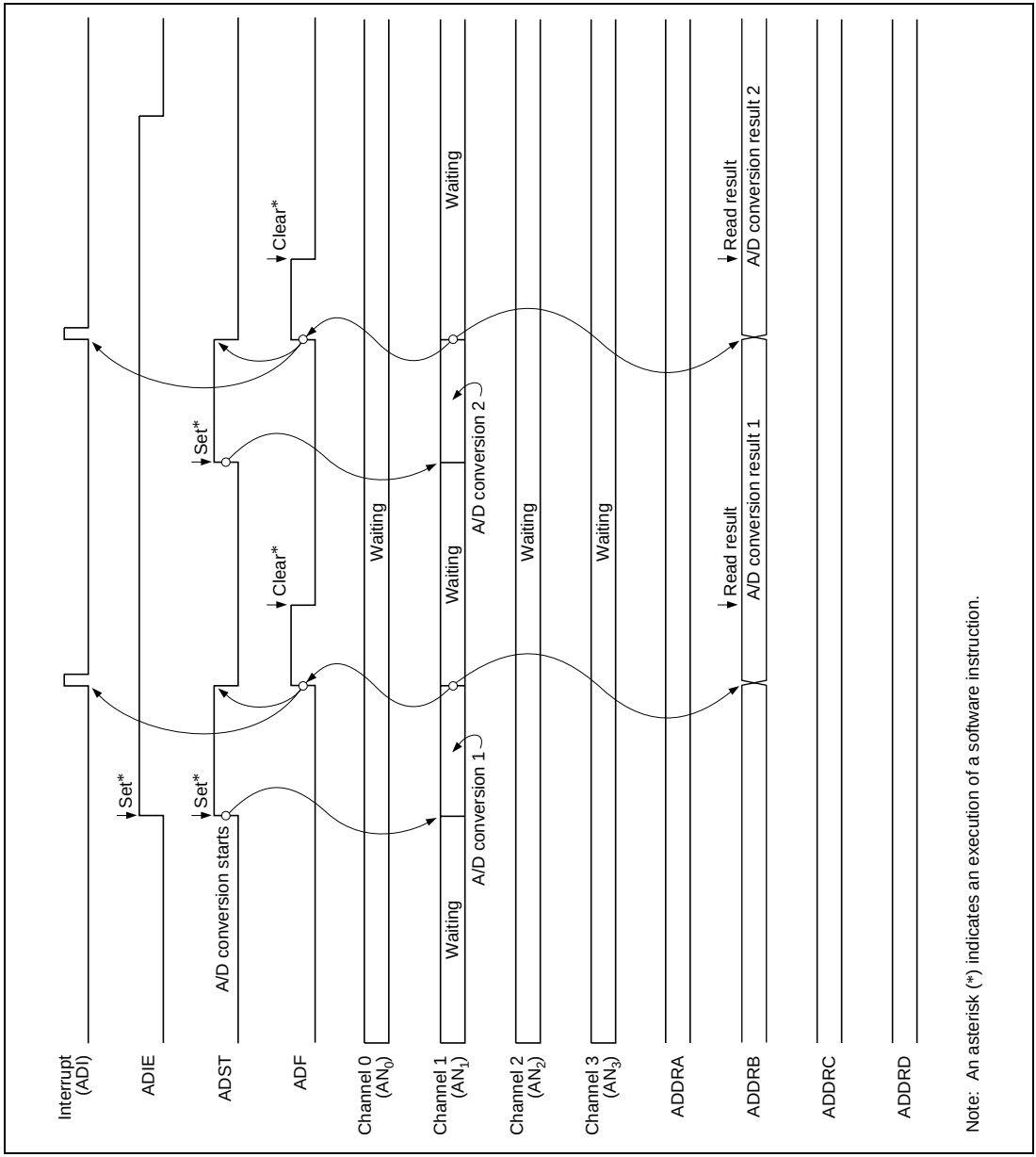


Figure 13.3 A/D Operation in Single Mode (When Channel 1 is Selected)

13.4.2 Scan Mode

The scan mode can be used to monitor analog inputs on one or more channels. When the ADST bit is set to 1 by software or an external trigger input, A/D conversion starts from the first channel selected by the CH bits. When CH2 = 0, the first channel is AN_0. When CH2 = 1, the first channel is AN_4.

If the scan group includes more than one channel (i.e., if bit CH1 or CH0 is set), the conversion of the next channel begins as soon as the conversion of the first channel ends.

Conversion of the selected channels continues cyclically until the ADST bit is cleared to 0. The conversion results are stored in the data registers corresponding to the selected channels.

Before selecting scan mode, clock, and analog input channels, the program should clear the ADST bit to 0 to ensure that the A/D converter has stopped. Changing the mode, clock, or channel selection while the A/D conversion is in progress can generate conversion errors.

The following is an example of the A/D conversion process when three channels in group 0 are selected (AN_0, AN_1, and AN_2). See figure 13.4.

1. The program clears the ADST bit to 0, selects scan mode (SCAN = 1), scan group 0 (CH2 = 0), and analog input channels AN_0 to AN_2 (CH1 and CH0 = 0), then sets the ADST bit to 1 to start the A/D conversion.

   Coding Example: (with slow clock and ADI interrupt enabled)

   ```
   BCLR  #5,@H'FE88
   MOV.B #H'72,@FE88
   ```

2. The A/D converter samples the input at AN_0, converts the voltage level to a digital value, and transfers the result to register ADDRA.
3. Next, the A/D converter samples and converts the input at AN_1 and transfers the result to ADDRB. Then it does the same for AN_2 and transfers the result to ADDRC.
4. After all selected channels (AN_0 to AN_2) have been converted, the A/D converter sets the ADF bit to 1. If the ADIE bit is set to 1, an A/D interrupt (ADI) is requested. Otherwise, the A/D converter begins converting from AN_0 again.
5. Steps (2) to (4) are repeated cyclically as long as the ADST bit remains set at 1.

To stop the A/D converter, the program must clear the ADST bit to 0. Note that the program must read the conversion data before stopping the A/D converter. If not, the data in the A/D data registers may change when clearing the ADST bit to 0.

Regardless of which channel is being converted when the ADST bit is cleared to 0, the moment the ADST bit is set to 1 again, conversion begins from the first selected channel (AN_0 or AN_4).

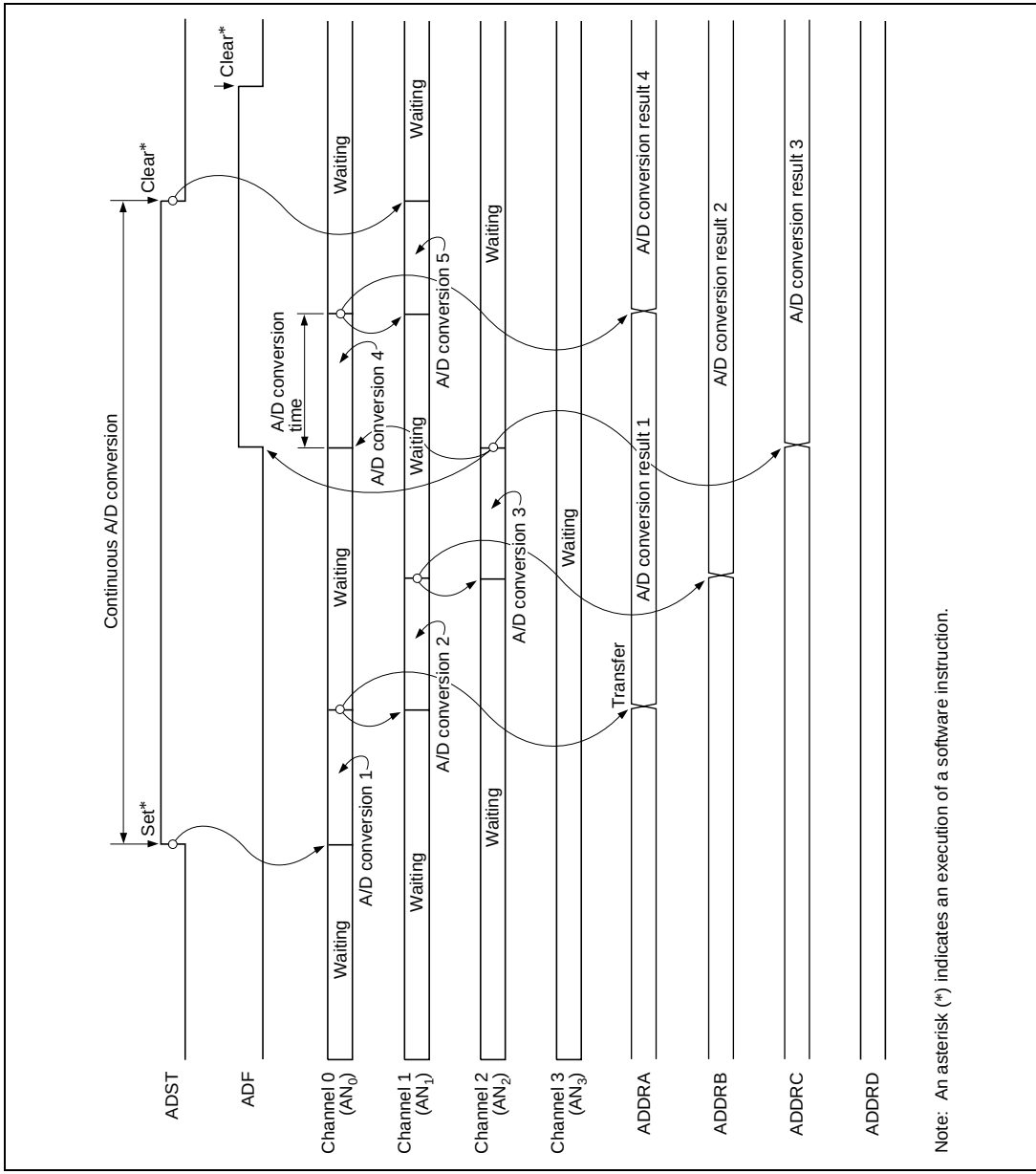


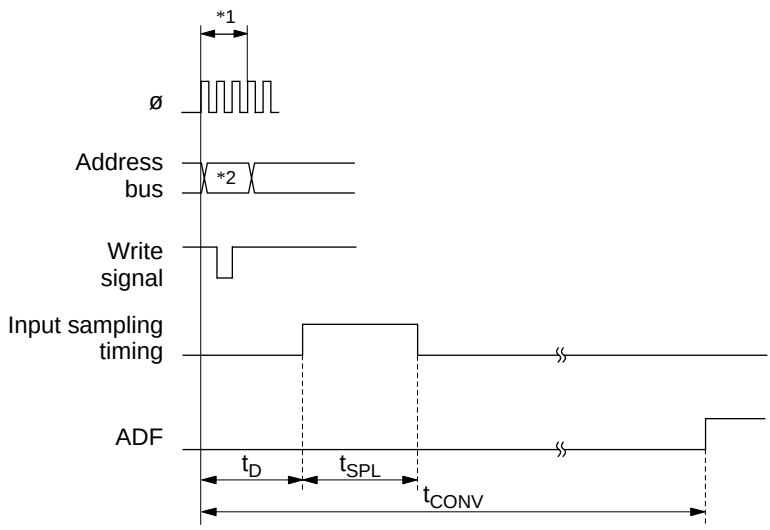
Figure 13.4 A/D Operation in Scan Mode (When Channels 0 or 2 are Selected)

13.4.3 Input Sampling Time and A/D Conversion Time

The A/D converter includes a built-in sample-and-hold circuit. Sampling of the input starts at time t_D after the ADST bit is set to 1. The sampling process lasts for a period of t_{SPL}. The actual A/D conversion begins after the sampling is completed. See figure 13.5 and table 13.4.

The total conversion time includes t_D and t_{SPL}. The purpose of t_D is to synchronize the ADCSR write time with the A/D conversion process, therefore the duration of t_D is variable. As a result, the total conversion time varies within minimum to maximum ranges.

In the scan mode, the ranges given in table 13.4 apply to the first conversion. The duration of the second and subsequent conversion processes is fixed at 256 states (CKS = 0) or 128 states (CKS = 1).



Notes: 1. ADCSR write cycle
2. ADCSR address
t_D: Synchronization delay
t_{SPL}: Input sampling time
t_{CONV}: Total A/D conversion time

Figure 13.5 A/D Conversion Timing

Table 13.4 A/D Conversion Time (Single Mode)

		CKS = 0			CKS = 1		
Item	**Symbol**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**
Synchronization delay	t_D	10	—	17	6	—	9
Input sampling time	t_{SPL}	—	80	—	—	40	—
Total A/D conversion time	t_{CONV}	259	—	266	131	—	134

Note: Values are in number of states.

13.4.4 A/D Conversion Start by External Trigger Input

The A/D converter can be started when an external trigger is input.

The external trigger is input from the $\overline{\text{ADTRG}}$ input pin when the trigger enable (TRGE) bit in the A/D control register (ADCR) is set to 1. When the $\overline{\text{ADTRG}}$ input pin is asserted low, the A/D start (ADST) bit in the A/D control/status register (ADCSR) is set to 1 after 1.5 to 2 cycles and the A/D conversion begins (figure 13.6).

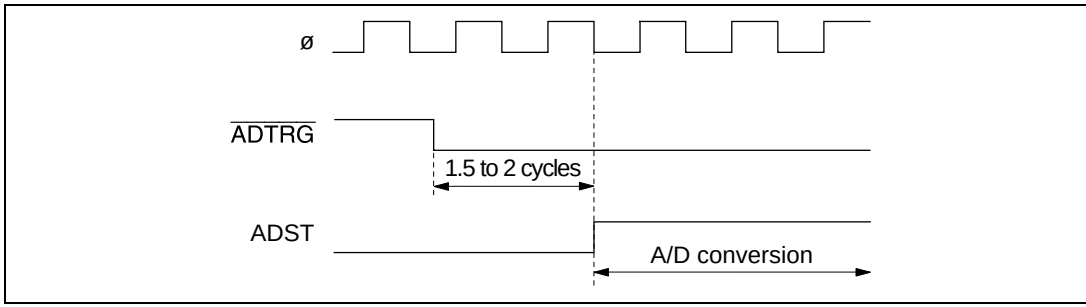


Figure 13.6 External Trigger Input Timing

13.5 Interrupts and the Data Transfer Controller

The ADI interrupt request is enabled or disabled by the ADIE bit in ADCSR.

When the ADI bit in data transfer enable register DTED (bit 0 at address H'FFF7) is set to 1, the ADI interrupt is served by the data transfer controller. The DTC can be used to transfer A/D results to a buffer in memory or to an I/O port. The DTC will automatically clear the ADF bit to 0.

Note: In scan mode, the DTC can transfer data for only one channel per interrupt, even if two or more channels are selected.

14.1 Overview

The H8/570 includes 2 kbytes of on-chip static RAM connected to the CPU by a 16-bit data bus. Both byte and word access to RAM are performed in two states, enabling rapid data transfer and instruction execution.

The addresses of RAM are assigned at H'F680 to H'FE7F in the MCU's address space. A RAM control register (RAMCR) can enable or disable RAM, thus permitting these addresses to be allocated for external memory instead.

14.1.1 Block Diagram

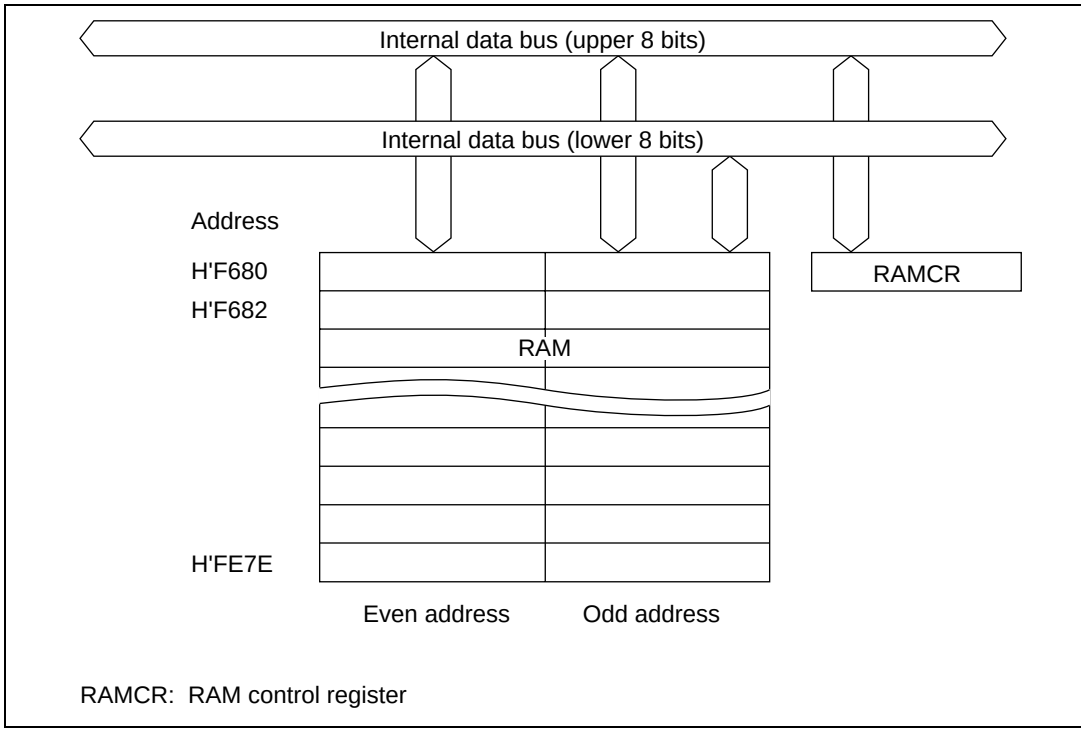


Figure 14.1 Block Diagram of RAM

14.1.2 Register Configuration

Table 14.1 RAM Control Register

Name	Abbreviation	R/W	Initial Value	Address
RAM control register	RAMCR	R/W	H'FF	H'FF49

14.2 RAM Control Register (RAMCR)

Bit	7	6	5	4	3	2	1	0
	RAME	IBE	—	—	—	—	—	—
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	—	—	—	—	—	—

The RAM control register (RAMCR) is an 8-bit register that enables or disables RAM.

Bit 7—RAM Enable (RAME): This bit enables or disables RAM.

The RAME bit is initialized to 1 on the rising edge of the reset signal. It is not initialized in the software standby mode.

Bit 7 RAME	Description
0	RAM is disabled.
1	RAM is enabled. (Also, the initial value.)

Bit 6—ISP Bus Enable (IBE): This bit enables or disables the execution of the ISP memory access function.

The IBE bit is also initialized to 1 at reset. It is cleared to 0 by an address error generated by the ISP.

Bit 6 IBE	Description
0	Bus access right of the ISP is disabled.
1	Bus access right of the ISP is enabled. (Also, the initial value.)

Bits 5 to 0—Reserved: These bits cannot be modified and are always read as 1.

14.3 Operation

If the RAME bit is set to 1, access to addresses H'F680 to H'FE7F is directed to RAM. If the RAME bit is cleared to 0, access to these addresses will be directed to the external data bus.

15.1 PROM Mode

15.1.1 PROM Mode Setup

The PROM mode of the H8/570 allows the ISP µROM to be programmed while halting its microcomputer functions.

To select the PROM mode, apply the signal inputs listed in table 15.1.

Table 15.1 Selection of PROM Mode

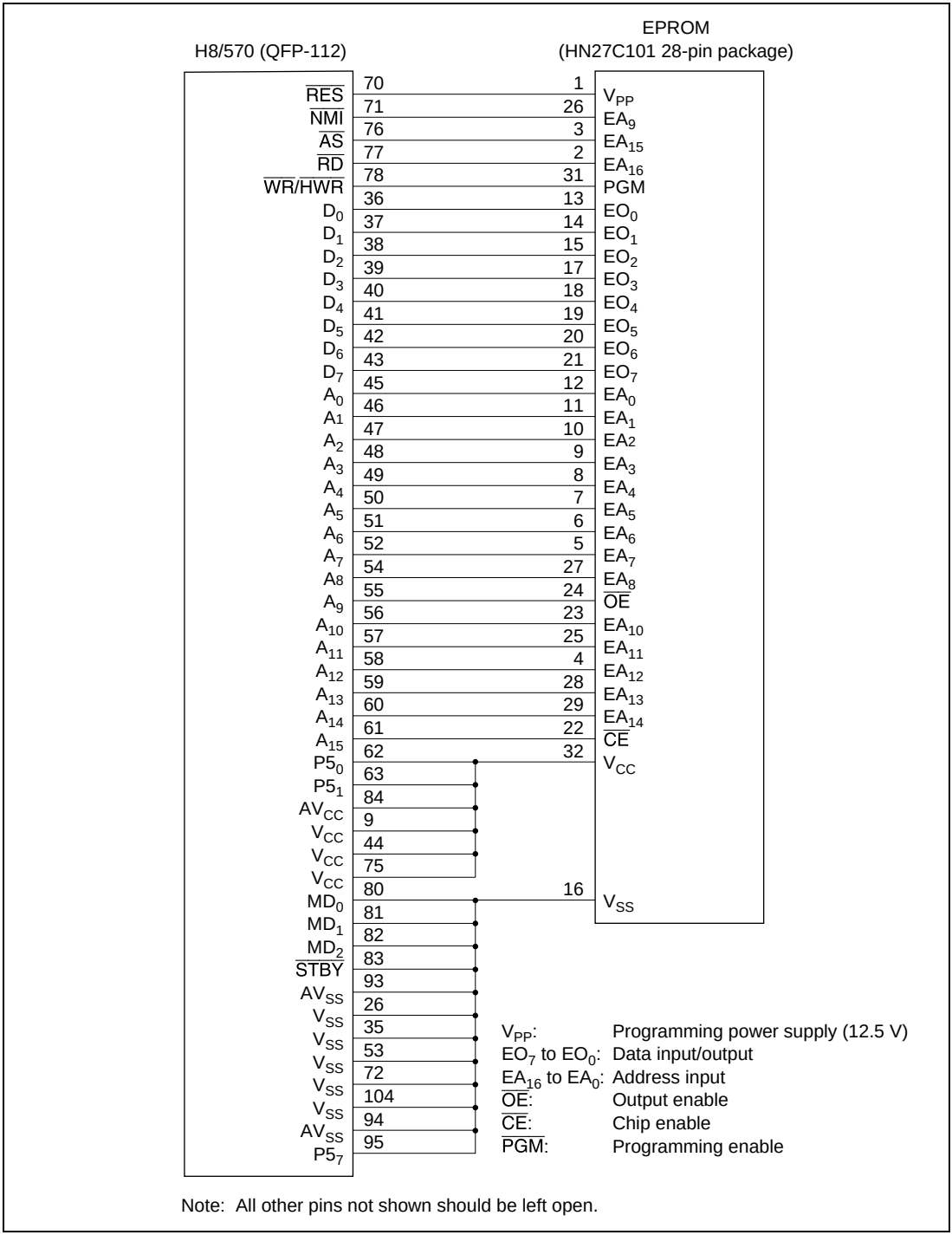
Pin	Input
Mode pins (MD_2, MD_1, and MD_0)	Low
$\overline{STBY}$ pin	Low
$P5_7$	Low
$P5_1$ and $P5_0$	High

15.1.2 Socket Adapter Pin Assignments and Memory Map

The H8/570 can be programmed with a general-purpose PROM writer by attaching a socket adapter (table 15.2). The socket adapter converts the pin assignments of the H8/570 to the HN27C101 EPROM. See figure 15.1.

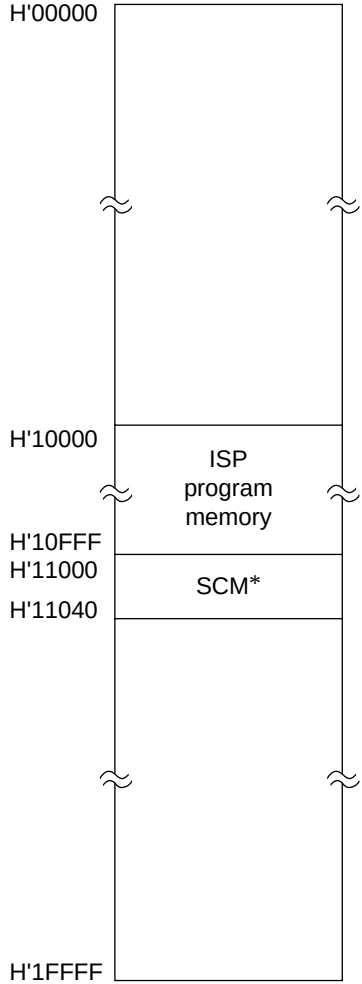
Table 15.2 Socket Adapter

Package	Socket Adapter
112-pin plastic QFP (QFP-112)	HS578ESH01H



Note: All other pins not shown should be left open.

Figure 15.1 Socket Adapter Pin Assignments



Note: * The upper 2 bits must be set to 1.

Figure 15.2 Memory Map in PROM Mode

15.2 Programming

The write, verify, and inhibited submodes of the PROM mode are selected by pins $\overline{CE}$, $\overline{OE}$, and $\overline{PGM}$ (table 15.3).

Table 15.3 Selection of Submodes in PROM Mode

	Pins				
Mode	$\overline{CE}$	$\overline{OE}$	$\overline{PGM}$	O_7 to O_0	A_{16} to A_0
Write	Low	High	Low	Data input	Address input
Verify	Low	Low	High	Data output	Address input
Programming inhibited	Low	Low	Low	High impedance	Address input
	Low	High	High	High impedance	Address input
	High	Low	Low	High impedance	Address input
	High	High	High	High impedance	Address input

Note: The V_{PP} and V_{CC} pins must be held at V_{PP} and V_{CC} voltage levels.

The PROM of the H8/570 uses the same standard read/write specifications as the HN27C101 EPROM. Since the H8/570 does not support the page programming procedure, the page programming mode cannot be used.

15.2.1 Writing and Verifying

An efficient, high-speed programming procedure can be used to write and verify PROM data. See figure 15.3. This procedure writes data quickly without subjecting the chip to voltage stress and without sacrificing data reliability. Data H'FF is written into all unused addresses.

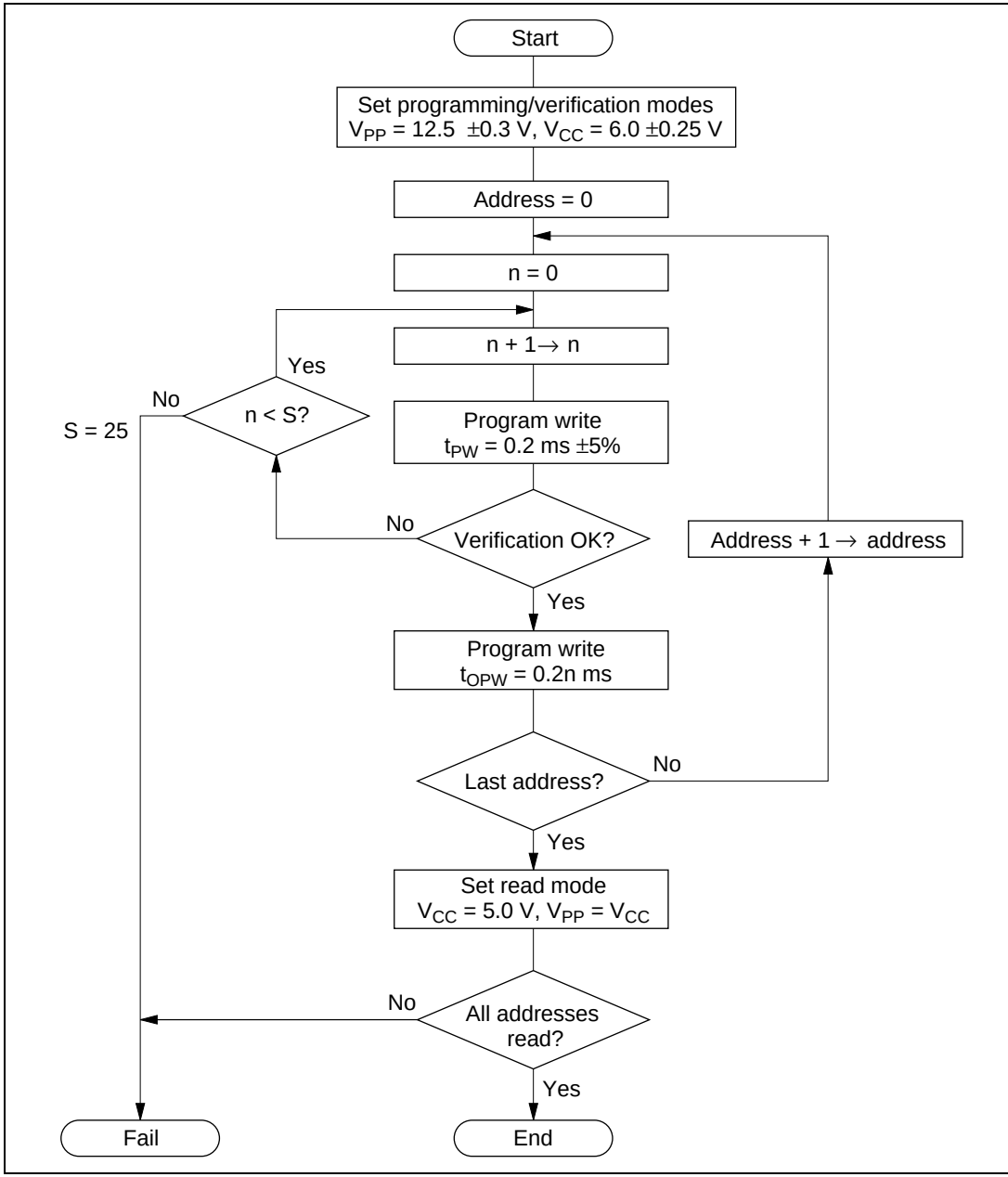


Figure 15.3 High-Speed Programming Flowchart

Table 15.4 DC Characteristics (V_{CC} = 6.0 V ±0.25 V, V_{PP} = 12.5 V ±0.3 V, V_{SS} = 0 V, T_a = 25°C ±5°C)

Item		Symbol	Min	Typ	Max	Unit	Measurement Conditions
Input high voltage	O_7 to O_0, A_{16} to A_0, $\overline{OE}$, $\overline{CE}$, $\overline{PGM}$	V_{IH}	2.4	—	V_{CC} + 0.3	V	
Input low voltage	O_7 to O_0, A_{16} to A_0, $\overline{OE}$, $\overline{CE}$, $\overline{PGM}$	V_{IL}	–0.3	—	0.8	V	
Output high voltage	O_7 to O_0	V_{OH}	2.4	—	—	V	I_{OH} = –200 µA
Output low voltage	O_7 to O_0	V_{OL}	—	—	0.45	V	I_{OL} = 1.6 mA
Input leakage current	O_7 to O_0, A_{16} to A_0, $\overline{OE}$, $\overline{CE}$, $\overline{PGM}$	$\|I_{IL}\|$	—	—	2	µA	V_{in} = 5.25 V/ 0.5 V
V_{CC} current		I_{CC}	—	—	40	mA	
V_{PP} current		I_{PP}	—	—	40	mA	

Table 15.5 AC Characteristics (V_{CC} = 6.0 V ±0.25 V, V_{PP} = 12.5 V ±0.3 V, T_a = 25°C ±5°C)

Item	Symbol	Min	Typ	Max	Unit	Measurement Conditions
Address setup time	t_{AS}	2	—	—	µs	Figure 15.4*
$\overline{OE}$ setup time	t_{OES}	2	—	—	µs	
Data setup time	t_{DS}	2	—	—	µs	
Address hold time	t_{AH}	0	—	—	µs	
Data hold time	t_{DH}	2	—	—	µs	
Data output disable time	t_{DF}	—	—	130	ns	
V_{PP} setup time	t_{VPS}	2	—	—	µs	
Program pulse width	t_{PW}	0.19	0.20	0.21	ms	
$\overline{OE}$ pulse width for overprogramming	t_{OPW}	0.19	—	5.25	ms	
V_{CC} setup time	t_{VCS}	2	—	—	µs	
$\overline{CE}$ setup time	t_{CES}	2	—	—	µs	
Data output delay time	t_{OE}	0	—	150	ns	

Notes: Input pulse level: 0.8 V to 2.2 V
Input rise/fall time ≤ 20 ns
Timing reference levels: Input—1.0 V, 2.0 V; Output—0.8 V, 2.0 V

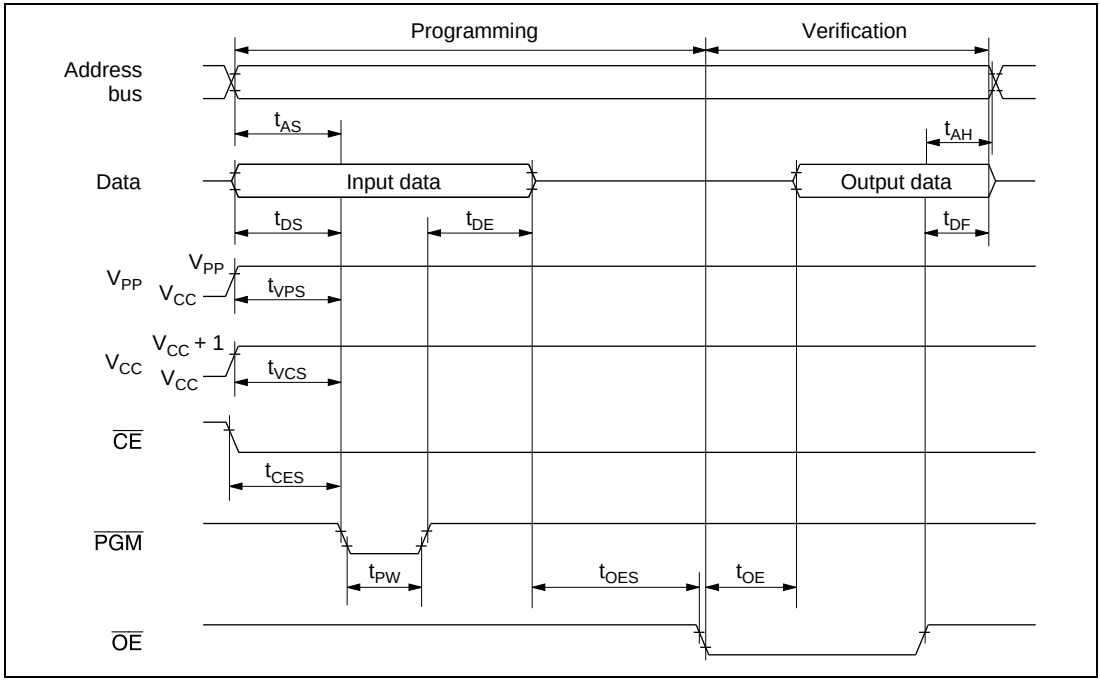


Figure 15.4 PROM Programming/Verification Timing

15.2.2 Reliability of Written Data

An effective way to assure the data-hold characteristics of the programmed MCU is to bake it at 150°C, then screen it for data errors. This procedure (figure 15.5) quickly eliminates chips with defective PROM memory cells.

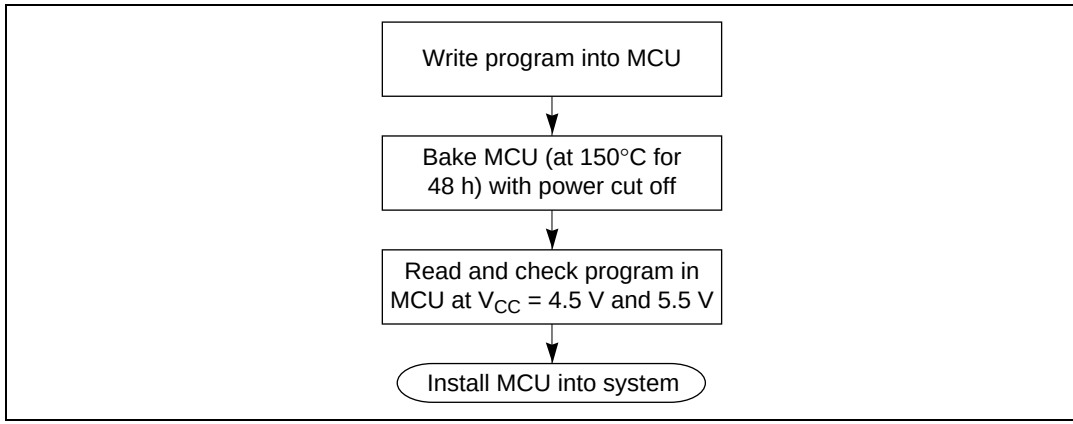

Write program into MCU
Bake MCU (at 150°C for 48 h) with power cut off
Read and check program in MCU at V_{CC} = 4.5 V and 5.5 V
Install MCU into system

Figure 15.5 Recommended Screening Procedure

If a series of write errors occur while the same PROM writer is in use, stop programming and check the PROM writer and socket adapter for defects.

Contact a Hitachi representative in the case of any abnormal conditions noted during programming or in the screening of program data after high-temperature baking.

Section 16 Power-Down State

16.1 Overview

The H8/570 has a power-down state that greatly reduces the power consumption by stopping the CPU functions. This power-down state includes three modes.

- Sleep mode: A software-triggered mode in which only the CPU halts with the rest of the MCU remaining active
- Software standby mode: A software-triggered mode in which the entire chip is inactive
- Hardware standby mode: A hardware-triggered mode in which the entire chip is inactive

The sleep mode and software standby mode are entered from the program execution state by executing the SLEEP instruction under the conditions of table 16.1. The hardware standby mode is entered from any other state by asserting the $\overline{\text{STBY}}$ pin low.

The conditions for entering and leaving a mode, the statuses of the clock, CPU, on-chip supporting modules, etc., differs for each power-down mode (table 16.1).

Table 16.1 Power-Down State

Mode	To Enter Mode	Clock	CPU	CPU Registers	Support Modules	RAM	I/O Ports	To Exit Mode
Sleep mode	Execute SLEEP instruction	Running	Halt	Data held	Running	Data held	Unchanged	• An interrupt • $\overline{\text{RES}}$ low • $\overline{\text{STBY}}$ low
Software standby mode	Set SSBY bit in SBYCR to 1, then execute SLEEP instruction*	Halt	Halt	Data held	Halt and partially initialized	Data held	Unchanged	• $\overline{\text{RES}}$ low • $\overline{\text{STBY}}$ low
Hardware standby mode	Assert $\overline{\text{STBY}}$ low	Halt	Halt	Initialized	Halt	Data held	High-impedance state	• $\overline{\text{STBY}}$ high, then $\overline{\text{RES}}$ low → high

Notes: * The watchdog timer must also be stopped.
SBYCR: Software standby control register
SSBY: Software standby bit

16.2 Sleep Mode

16.2.1 Entering Sleep Mode

The execution of the SLEEP instruction causes a transition from the program execution state to the sleep mode. After executing the SLEEP instruction, the CPU halts, but the contents of its internal registers remain unchanged. The functions of the on-chip supporting modules do not stop in the sleep mode.

16.2.2 Exiting Sleep Mode

The MCU wakes up from the sleep mode when it receives an internal or external interrupt request, or a low input at the $\overline{\text{RES}}$ or $\overline{\text{STBY}}$ pin.

Wake-Up by Interrupt: An interrupt releases the sleep mode and starts either the CPU's interrupt-handling sequence or the data transfer controller (DTC).

If the interrupt is served by the DTC, the CPU executes the instruction following the SLEEP instruction after the data transfer is completed, unless the count in the data transfer count register (DTCR) is 0.

If an interrupt with a mask level equal to or less than the level in the CPU's status register (SR) is requested, the interrupt is left pending while in the sleep mode. Also, if an interrupt from an on-chip supporting module is disabled by the corresponding enable/disable bit in the module's control register, the interrupt cannot be requested to wake the MCU out of sleep mode.

Wake-Up by $\overline{\text{RES}}$ Pin: When the $\overline{\text{RES}}$ pin is asserted low, the MCU exits from the sleep mode and enters the reset state.

Wake-Up by $\overline{\text{STBY}}$ Pin: When the $\overline{\text{STBY}}$ pin is asserted low, the MCU exits from the sleep mode and enters the hardware standby mode.

16.3 Software Standby Mode

16.3.1 Entering Software Standby Mode

To enter the software standby mode, the program must set the standby bit (SSBY) in the software standby control register (SBYCR) to 1, then execute the SLEEP instruction. See table 16.2.

Table 16.2 Software Standby Control Register

Name	Abbreviation	R/W	Initial Value	Address
Software standby control register	SBYCR	R/W	H'7F	H'FF4B

In the software standby mode, the CPU, clock, and the on-chip supporting modules functions all stop, reducing power consumption to an extremely low level. Thus, resetting the on-chip supporting modules and their registers to their initial state. By maintaining a minimum voltage supply of at least 2 V, the contents of the CPU registers and RAM remain unchanged. The I/O ports also remain in their current states.

16.3.2 Software Standby Control Register (SBYCR)

Bit	7	6	5	4	3	2	1	0
	SSBY	—	—	—	—	—	—	—
Initial value	0	1	1	1	1	1	1	1
Read/Write	R/W	—	—	—	—	—	—	—

The software standby control register (SBYCR) is an 8-bit register that controls the action of the SLEEP instruction.

Bit 7—Software Standby (SSBY): This bit enables the transition to either the sleep mode or software standby mode.

Bit 7 SSBY	Description
0	The SLEEP instruction causes a transition to the sleep mode. (Also, the initial value.)
1	The SLEEP instruction causes a transition to the software standby mode.

The watchdog timer must be stopped before the MCU can enter the software standby mode. To stop the watchdog timer, clear the timer enable bit (TME) in the watchdog timer's timer control/status register (TCSR) to 0. The SSBY bit cannot be set to 1 while the TME bit is set to 1.

This bit is cleared to 0 by reset or by a transition to the hardware standby mode.

Bits 6 to 0—Reserved: These bits cannot be modified and are always read as 1.

16.3.3 Exiting Software Standby Mode

The MCU can be brought out of the software standby mode by an input at the $\overline{\text{RES}}$ pin or $\overline{\text{STBY}}$ pin.

Recovery by $\overline{\text{RES}}$ Pin: When the $\overline{\text{RES}}$ pin is asserted low, the clock oscillator starts. And when the $\overline{\text{RES}}$ pin is returned high, the CPU begins executing the reset sequence.

When the MCU recovers from the software standby mode by reset, the clock pulses are immediately supplied to the MCU. Be sure to hold the $\overline{\text{RES}}$ pin low long enough for the clock to stabilize.

Recovery by $\overline{\text{STBY}}$ Pin: When the $\overline{\text{STBY}}$ pin is asserted low, the MCU exits from the software standby mode and enters the hardware standby mode.

16.3.4 Application Notes

The I/O ports remain in their current states in the software standby mode. If the output of a port is high, the output current will not be reduced in the software standby mode.

16.4 Hardware Standby Mode

16.4.1 Entering Hardware Standby Mode

Regardless of its current state, the MCU enters the hardware standby mode whenever the $\overline{\text{STBY}}$ pin is asserted low.

This hardware standby mode reduces the power consumption drastically by halting the CPU, stopping all functions of the on-chip supporting modules, and placing the I/O ports in the high-impedance state.

The registers of the on-chip supporting modules are reset to their initial values. Only RAM remains unchanged, provided the minimum voltage supply (2 V) is maintained.*

Note: * The RAME bit in the RAM control register should be cleared to 0 before the $\overline{\text{STBY}}$ pin is asserted low, in order to disable RAM during the hardware standby mode.

16.4.2 Exiting Hardware Standby Mode

Recovery from the hardware standby mode requires inputs at both the $\overline{\text{STBY}}$ and $\overline{\text{RES}}$ pins.

When the $\overline{\text{STBY}}$ pin is driven high, the clock oscillator begins running. The $\overline{\text{RES}}$ pin should be held low at this time just long enough for the clock to stabilize. Then, when the input of the $\overline{\text{RES}}$ pin changes from low to high, the reset sequence is executed and the MCU returns to the program execution state.

16.4.3 Timing Sequence of Hardware Standby Mode

The sequence (figure 16.1) for entering the hardware standby mode is done by first asserting the $\overline{\text{RES}}$ pin low to place the MCU in the reset state, then asserting the $\overline{\text{STBY}}$ pin low to enter the hardware standby mode which stops the clock. To exit this mode (figure 16.1), the $\overline{\text{STBY}}$ pin must be driven high with enough delay to stabilize the clock before driving the $\overline{\text{RES}}$ pin high.

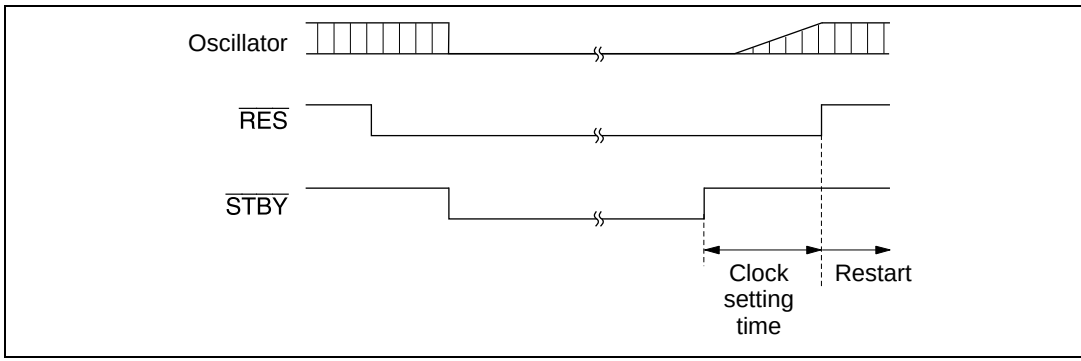


Figure 16.1 Hardware Standby Sequence

Section 17 E-Clock Interface

17.1 Overview

The H8/570 can generate an E-clock output for interfacing with E-clock-based peripheral devices. Special instructions (MOVTPE, MOVFPE) perform data transfers synchronized with the E clock.

The E clock is created by dividing the system clock (ø) by 8. The E clock is output at the $P5_6$ pin when the $P5_6$DDR bit in the port 5 data direction register (P5DDR) is set to 1.

When the CPU executes a special instruction that synchronizes with the E clock, the address is output on the address bus as usual, but the data bus and the $\overline{RD}$, $\overline{WR}$, $\overline{HWR}$, and $\overline{LWR}$ signal lines do not become active until the falling edge of the E clock is detected. The length of the access cycle for an instruction synchronized with the E clock is variable, accordingly. See figures 17.1 and 17.2.

The wait state controller (WSC) does not insert any wait states (T_w) during the execution of an instruction that is synchronized with the E clock.

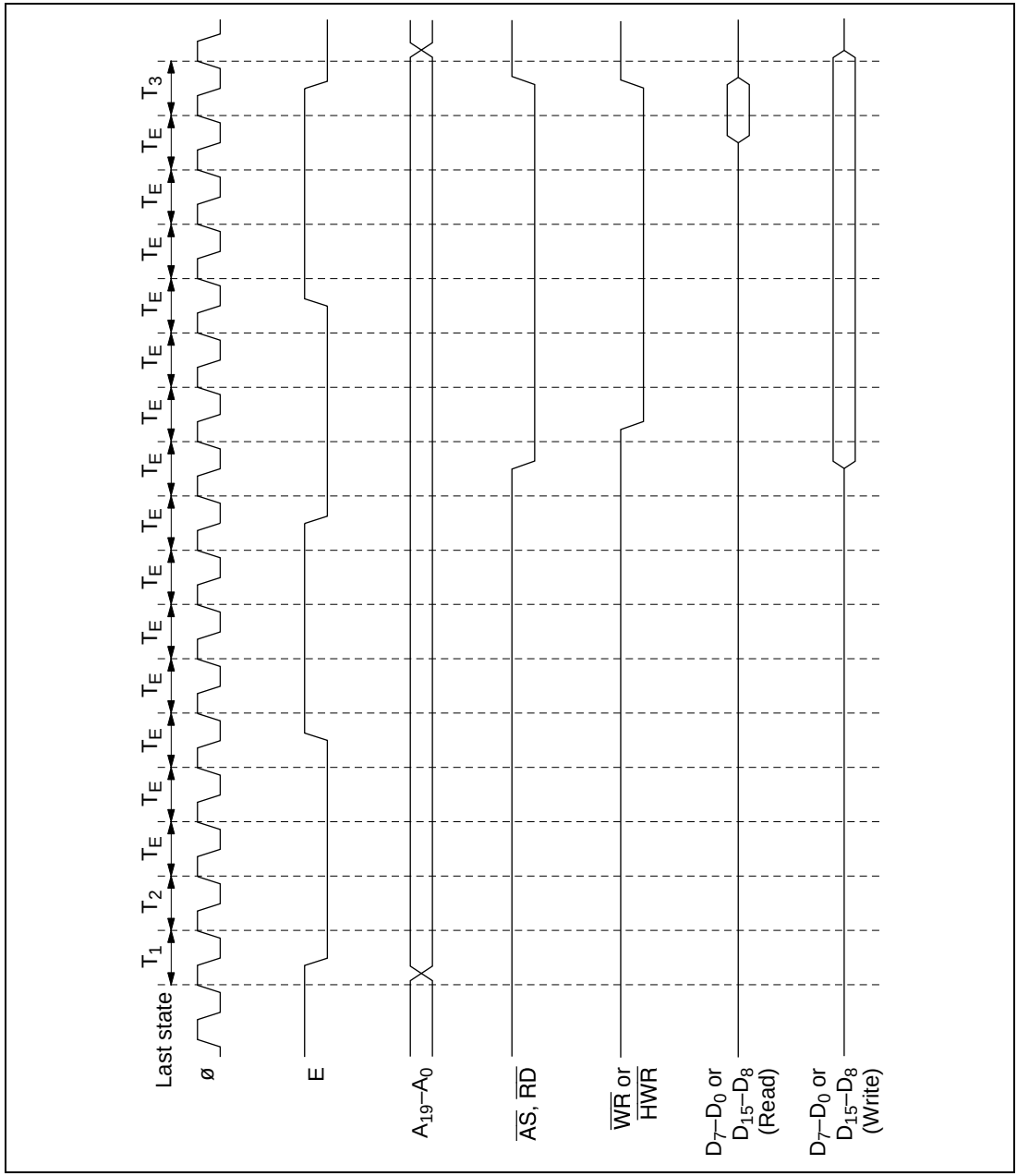


Figure 17.1 Execution Cycle of Instruction Synchronized with E Clock in Expanded Modes (Maximum Synchronization Delay)

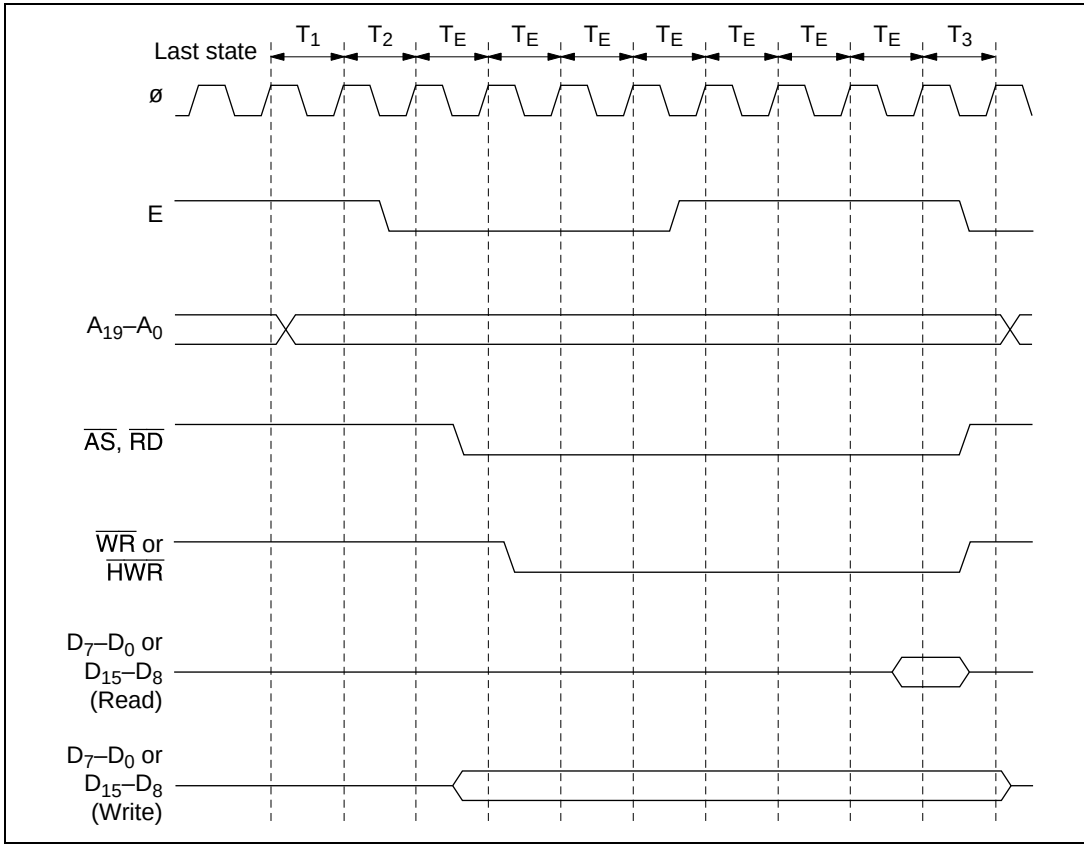


Figure 17.2 Execution Cycle of Instruction Synchronized with E Clock in Expanded Modes (Minimum Synchronization Delay)

Section 18 Electrical Specifications

18.1 Absolute Maximum Ratings

Item		Symbol	Rating		Unit
Supply voltage		V_{CC}	–0.3 to +7.0		V
Programming voltage		V_{PP}	–0.3 to +13.5		V
Input voltage	(Except port 7)	V_{in}	–0.3 to V_{CC} + 0.3		V
	(Port 7)	V_{in}	–0.3 to AV_{CC} + 0.3		V
Analog supply voltage		AV_{CC}	–0.3 to +7.0		V
Analog input voltage		V_{AN}	–0.3 to AV_{CC} + 0.3		V
Operating temperature		T_{opr}	Standard:	–20 to +75 (f = 0.5–10 MHz)	°C
				–20 to +60 (f = 0.5–12 MHz)	°C
			Wide-range:	–40 to +85 (f = 0.5–10 MHz)	°C
Storage temperature		T_{stg}	–55 to +125		°C

Note: If these maximum ratings are exceeded, the chip may become permanently damaged. Normal operation should perform slightly below the recommended operating conditions.

18.2 Electrical Characteristics

18.2.1 DC Characteristics

Table 18.1 DC Characteristics

Conditions: $V_{SS} = AV_{SS} = 0$ V; For f = 8 or 10 MHz: $V_{CC} = 5.0$ V ±10%, T_a = –20 to +75°C (Standard) or –40 to +85°C (Wide-Range), *$AV_{CC} = 5.0$ V ±10%; For f = 12 MHz: $V_{CC} = 5.0$ V (–5% to +10%), T_a = –20 to +60°C (Standard), *$AV_{CC} = 5.0$ V (–5% to +10%)

Item		Symbol	Min	Typ	Max	Unit	Measurement Conditions
Input high voltage	$\overline{RES}$, $\overline{STBY}$, MD_2, MD_1, MD_0	V_{IH}	V_{CC} – 0.7	—	V_{CC} +0.3	V	
	EXTAL		0.7V_{CC}	—	V_{CC} + 0.3	V	
	Port 7		2.0	—	AV_{CC} + 0.3	V	
	Other input pins		2.0	—	V_{CC} + 0.3	V	
Input low voltage	$\overline{RES}$, $\overline{STBY}$, MD_2, MD_1, MD_0	V_{IL}	–0.3	—	0.6	V	
	Other input pins		–0.3	—	0.8	V	
Input leakage current	$\overline{RES}$	| I_{in} |	—	—	10.0	µA	V_{in} = 0.5 to V_{CC} – 0.5 V
	$\overline{STBY}$, NMI, MD_2, MD_1, MD_0		—	—	1.0	µA	
	Port 7		—	—	1.0	µA	V_{in} = 0.5 to AV_{CC} – 0.5 V
Leakage current in tristate (off state)	All output pins	| I_{TSI} |	—	—	1.0	µA	V_{in} = 0.5 to V_{CC}– 0.5 V
Input pull-up MOS current	$P5_0$ to $P5_3$	$–I_P$	50	—	200	µA	V_{in} = 0 V
Output high voltage	All output pins	V_{OH}	V_{CC} – 0.5	—	—	V	I_{OH} = –200 µA
			3.5	—	—	V	I_{OH} = –1 mA
Output low voltage	$\overline{RES}$	V_{OL}	—	—	0.4	V	I_{OL} = 2.6 mA
	Other output pins		—	—	0.4	V	I_{OL} = 1.6 mA

Item		Symbol	Min	Typ	Max	Unit	Measurement Conditions
Input capacitance	$\overline{RES}$	C_{in}	—	—	80	pF	V_{in} = 0 V, f = 1 MHz, T_a = 25°C
	NMI		—	—	30	pF	
	All other input pins		—	—	20	pF	
Current dissipation*[1]	Normal operation	I_{CC}	—	50	80	mA	f = 8 MHz
			—	65	100	mA	f = 10 MHz
			—	80	120	mA	f = 12 MHz
	Sleep mode		—	42	70	mA	f = 8 MHz
			—	55	85	mA	f = 10 MHz
			—	70	100	mA	f = 12 MHz
	Standby		—	0.01	5.0	µA	$T_a \leq 50°C$
			—	—	20.0	µA	$50°C < T_a$
Analog supply current	During A/D conversion	AI_{CC}	—	1	2	mA	
	While waiting		—	0.01	5.0	µA	
RAM standby voltage		V_{RAM}	2.0	—	—	V	

Notes: *AV_{CC} must be connected to the power supply even if the A/D converter is not used.

1. $V_{IHmin} = V_{CC} – 0.5$ V, $V_{ILmax} = 0.5$ V, all output pins are in the no-load state, and all MOS input pull-ups are off.

Table 18.2 Allowable Output Current Sink Values

Conditions: $V_{SS} = AV_{SS} = 0$ V; For f = 8 or 10 MHz: $V_{CC} = 5.0$ V ±10%, T_a = –20 to +75°C (Standard) or –40 to +85°C (Wide-Range), $*AV_{CC} = 5.0$ V ±10%; For f = 12 MHz: $V_{CC} = 5.0$ V (–5% to +10%), T_a = –20 to +60°C (Standard), $*AV_{CC} = 5.0$ V (–5% to +10%)

Item		Symbol	Min	Typ	Max	Unit
Allowable output low current sink (per pin)	All output pins	I_{OL}	—	—	2.0	mA
Allowable output low current sink (total)	Total of all output pins	ΣI_{OL}	—	—	80	mA
Allowable output high current sink (per pin)	All output pins	$-I_{OH}$	—	—	2.0	mA
Allowable output high current sink (total)	Total of all output pins	$\Sigma (-I_{OH})$	—	—	25	mA

Note: To avoid degrading the reliability of the chip, be careful to not exceed the output current sink values of this table. In particular, when directly driving a Darlington transistor pair, be sure to insert a current-limiting resistor along the output path. See figure 18.1.

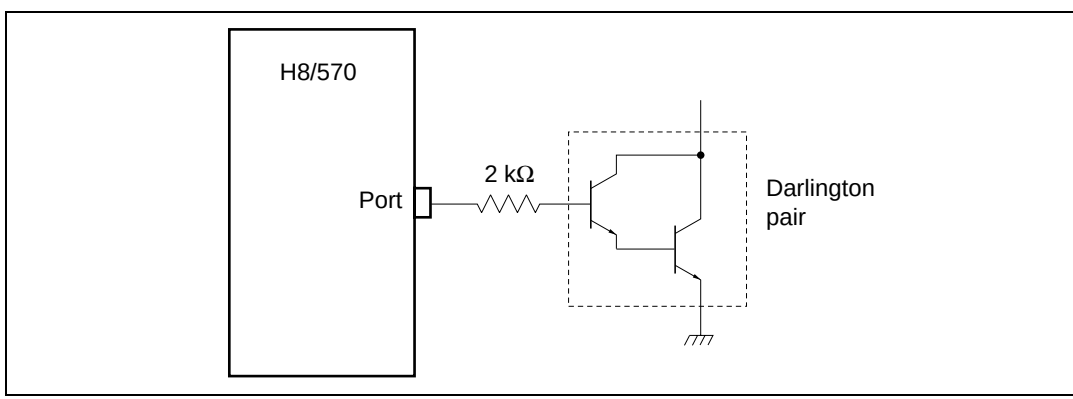


Figure 18.1 Example of Circuit for Driving a Darlington Transistor Pair

18.2.2 AC Characteristics

Table 18.3 Bus Timing

Conditions: V_{SS} = 0 V; For f = 0.5–10 MHz: V_{CC} = 5.0 V ±10%, T_a = –20 to +75°C (Standard) or –40 to +85°C (Wide-Range); For f = 0.5–12 MHz: V_{CC} = 5.0 V (–5% to +10%), T_a = –20 to +60°C

		8 MHz		10 MHz		12 MHz			
Item	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**	**Measurement Conditions**
Clock cycle time	t_{cyc}	125	2000	100	2000	83.3	2000	ns	Figure 18.4
Clock pulse width low	t_{CL}	45	—	35	—	25	—	ns	
Clock pulse width high	t_{CH}	45	—	35	—	25	—	ns	
Clock rise time	t_{Cr}	—	15	—	15	—	15	ns	
Clock fall time	t_{Cf}	—	15	—	15	—	15	ns	
Address delay time	t_{AD}	—	35	—	25	—	20*	ns	
Address hold time	t_{AH}	25	—	20	—	20	—	ns	
$\overline{AS}$ output delay time 1	t_{ASD1}	—	60	—	40	—	35	ns	
$\overline{AS}$ output delay time 2	t_{ASD2}	—	60	—	50	—	40	ns	
$\overline{RD}$ output delay time 1	t_{RDD1}	—	60	—	40	—	35	ns	
$\overline{RD}$ output delay time 2	t_{RDD2}	—	60	—	50	—	40	ns	
$\overline{WR}$ output delay time 1	t_{WRD1}	—	60	—	50	—	40	ns	
$\overline{WR}$ output delay time 2	t_{WRD2}	—	60	—	50	—	40	ns	
$\overline{WR}$ output delay time 3	t_{WRD3}	—	60	—	50	—	50	ns	Figure 18.6
Write data strobe pulse width	t_{DSWW}	150	—	120	—	100	—	ns	Figure 18.4
Address setup time 1	t_{AS1}	20	—	20	—	10	—	ns	
Address setup time 2	t_{AS2}	80	—	65	—	50	—	ns	
Address setup time 3	t_{AS3}	20	—	20	—	10	—	ns	Figure 18.6

Item	Symbol	8 MHz		10 MHz		12 MHz		Unit	Measurement Conditions
		Min	Max	Min	Max	Min	Max		
Read data setup time	t_{RDS}	35	—	25	—	15	—	ns	Figure 18.4
Read data hold time	t_{RDH}	0	—	0	—	0	—	ns	
Read data access time 1	t_{ACC1}	—	235	—	205	—	130	ns	
Read data access time 2	t_{ACC2}	—	100	—	90	—	75	ns	Figure 18.6
Write data delay time	t_{WDD}	—	65	—	60	—	55	ns	Figure 18.4
Write data setup time	t_{WDS}	15	—	10	—	10	—	ns	
Write data hold time	t_{WDH}	25	—	20	—	20	—	ns	
$\overline{\text{WAIT}}$ setup time	t_{WTS}	40	—	35	—	30	—	ns	Figure 18.5
$\overline{\text{WAIT}}$ hold time	t_{WTH}	10	—	10	—	10	—	ns	
Bus request setup time	t_{BRQS}	40	—	40	—	40	—	ns	Figure 18.11
Bus acknowledge delay time 1	t_{BACD1}	—	60	—	55	—	40	ns	
Bus acknowledge delay time 2	t_{BACD2}	—	60	—	55	—	40	ns	
Bus floating delay time	t_{BZD}	—	t_{BACD1}	—	t_{BACD1}	—	t_{BACD1}	ns	
E-clock delay time	t_{ED}	—	15	—	15	—	15	ns	Figure 18.12
E-clock rise time	t_{Er}	—	15	—	15	—	15	ns	
E-clock fall time	t_{Ef}	—	15	—	15	—	15	ns	
Read data hold time (E-clock sync)	t_{RDHE}	5	—	5	—	5	—	ns	Figure 18.7
Write data hold time (E-clock sync)	t_{WDHE}	40	—	30	—	20	—	ns	

Note: * 25 ns immediately after the bus is released.

Table 18.4 Control Signal Timing

Conditions: V_{SS} = 0 V; For f = 0.5–10 MHz: V_{CC} = 5.0 V ±10%, T_a = –20 to +75°C (Standard) or –40 to +85°C (Wide-Range); For f = 0.5–12 MHz: V_{CC} = 5.0 V (–5% to +10%), T_a = –20 to +60°C

		8 MHz		10 MHz		12 MHz			
Item	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Unit**	**Measurement Conditions**
$\overline{RES}$ setup time	t_{RESS}	200	—	200	—	200	—	ns	Figure 18.8
$\overline{RES}$ pulse width 1*	t_{RESW1}	6	—	6	—	6	—	t_{cyc}	
$\overline{RES}$ pulse width 2*	t_{RESW2}	520	—	520	—	520	—	t_{cyc}	
$\overline{RES}$ output delay time	t_{RESD}	—	100	—	100	—	100	ns	Figure 18.9
$\overline{RES}$ output pulse width	t_{RESOW}	132	—	132	—	132	—	t_{cyc}	
NMI setup time	t_{NMIS}	150	—	150	—	150	—	ns	Figure 18.10
NMI hold time	t_{NMIH}	10	—	10	—	10	—	ns	
$\overline{IRQ_0}$ setup time	t_{IRQ0S}	50	—	50	—	50	—	ns	
Crystal oscillator settling time (reset)	t_{OSC1}	20	—	20	—	20	—	ms	Figure 18.13

Note: * When the RSTOE bit in the reset control/status register (RSTCSR) is set to 1 or when the power is turned on, the RES pulse width is t_{RESW1}. When the RSTOE bit is cleared to 0, the RES pulse width is t_{RESW2}.

Table 18.5 Timing Conditions of On-Chip Supporting Modules

Conditions: V_{SS} = 0 V; For f = 0.5–10 MHz: V_{CC} = 5.0 V ±10%, T_a = –20 to +75°C (Standard) or –40 to +85°C (Wide-Range); For f = 0.5–12 MHz: V_{CC} = 5.0 V (–5% to +10%), T_a = –20 to +60°C

Item		Symbol	8 MHz		10 MHz		12 MHz		Unit	Measurement Conditions
			Min	Max	Min	Max	Min	Max		
Port	Output data delay time	t_{PWD}	—	100	—	100	—	100	ns	Figure 18.14
	Input data setup time	t_{PRS}	50	—	50	—	50	—	ns	
	Input data hold time	t_{PRH}	50	—	50	—	50	—	ns	
PWM	Timer output delay time	t_{PWOD}	—	100	—	100	—	100	ns	Figure 18.15
SCI	Input clock cycle (Async)	t_{Scyc}	2	—	2	—	2	—	t_{cyc}	Figure 18.16
	Input clock cycle (Sync)		4	—	4	—	4	—	t_{cyc}	
	Input clock pulse width	t_{SCKW}	0.4	0.6	0.4	0.6	0.4	0.6	t_{Scyc}	
	Transmit data delay time (Synchronous)	t_{TXD}	—	100	—	100	—	100	ns	Figure 18.17
	Receive data setup time (Synchronous)	t_{RXS}	100	—	100	—	100	—	ns	
	Receive data hold time (Synchronous)	t_{RXH}	100	—	100	—	100	—	ns	
ISP	Input data setup time	t_{ISPS}	50	—	50	—	50	—	ns	Figure 18.18
	Input data hold time	t_{ISPH}	50	—	50	—	50	—	ns	
	Output data delay time	t_{ISPD}	—	100	—	100	—	100	ns	Figure 18.19

Measurement Conditions for AC Characteristics

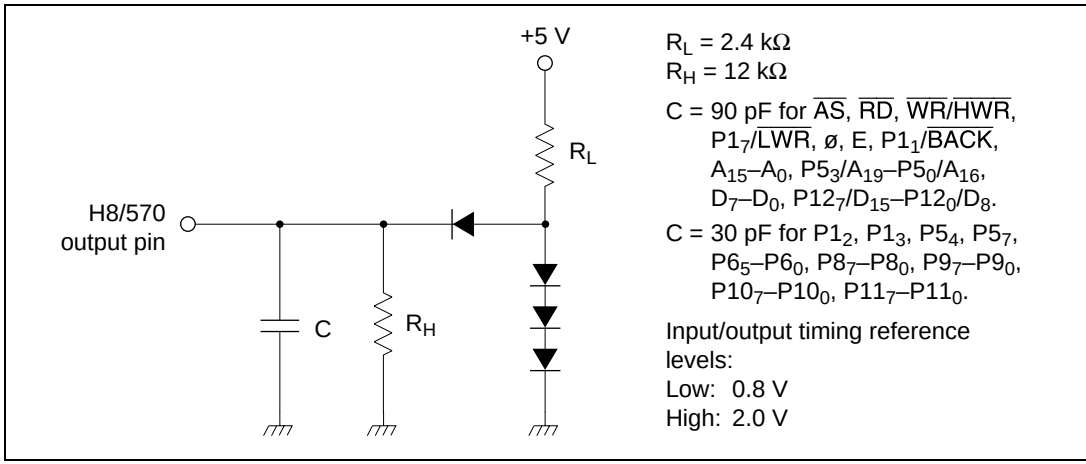

+5 V
R_L
H8/570
output pin
C
R_H
R_L = 2.4 kΩ
R_H = 12 kΩ
C = 90 pF for $\overline{AS}$, $\overline{RD}$, $\overline{WR}$/$\overline{HWR}$, $P1_7$/$\overline{LWR}$, ø, E, $P1_1$/$\overline{BACK}$, A_{15}–A_0, $P5_3$/A_{19}–$P5_0$/A_{16}, D_7–D_0, $P12_7$/D_{15}–$P12_0$/D_8.
C = 30 pF for $P1_2$, $P1_3$, $P5_4$, $P5_7$, $P6_5$–$P6_0$, $P8_7$–$P8_0$, $P9_7$–$P9_0$, $P10_7$–$P10_0$, $P11_7$–$P11_0$.
Input/output timing reference levels:
Low: 0.8 V
High: 2.0 V

Figure 18.2 Output Load Circuit

Reset Circuit

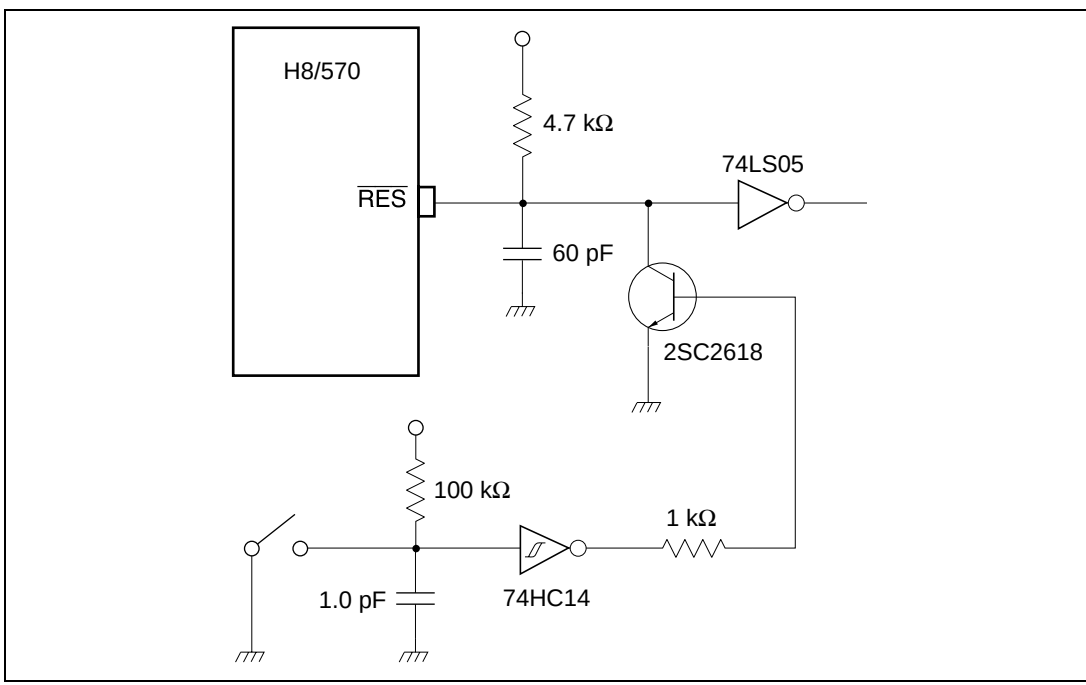


Figure 18.3 Example of Reset Circuit

18.2.3 A/D Converter Characteristics

Conditions: $V_{SS} = AV_{SS} = 0$ V; For f = 8 or 10 MHz: $V_{CC} = 5.0$ V ±10%, T_a = –20 to +75°C (Standard) or –40 to +85°C (Wide-Range), $*AV_{CC} = 5.0$ V ±10%; For f = 12 MHz: $V_{CC} = 5.0$ V (–5% to +10%), T_a = –20 to +60°C (Standard), $*AV_{CC} = 5.0$ V (–5% to +10%)

	8 MHz			10 MHz			12 MHz			
Item	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**
Resolution	10	10	10	10	10	10	10	10	10	Bits
Conversion time	—	—	16.75	—	—	13.4	—	—	11.17	µs
Analog input capacitance	—	—	20	—	—	20	—	—	20	pF
Allowable signal-source impedance	—	—	10	—	—	10	—	—	10	kΩ
Nonlinearity error	—	—	±2.0	—	—	±2.0	—	—	±2.0	LSB
Offset error	—	—	±2.0	—	—	±2.0	—	—	±2.0	LSB
Full-scale error	—	—	±2.0	—	—	±2.0	—	—	±2.0	LSB
Quantizing error	—	—	±0.5	—	—	±0.5	—	—	±0.5	LSB
Absolute accuracy	—	—	±2.5	—	—	±2.5	—	—	±2.5	LSB

18.3 MCU Operational Timing

18.3.1 Bus Timing

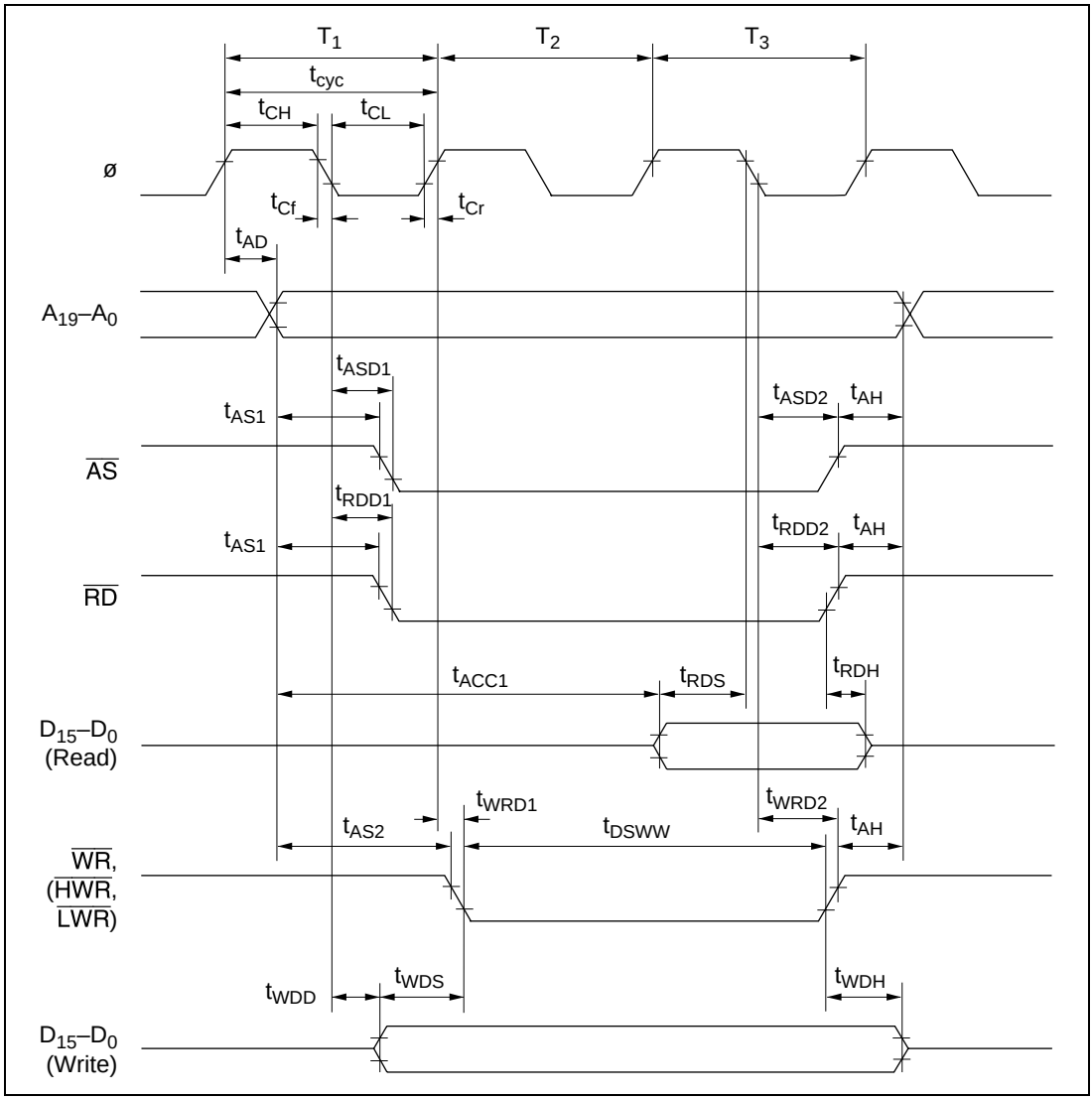


Figure 18.4 Basic Bus Cycle (Without Wait States) in Three-State Access Space

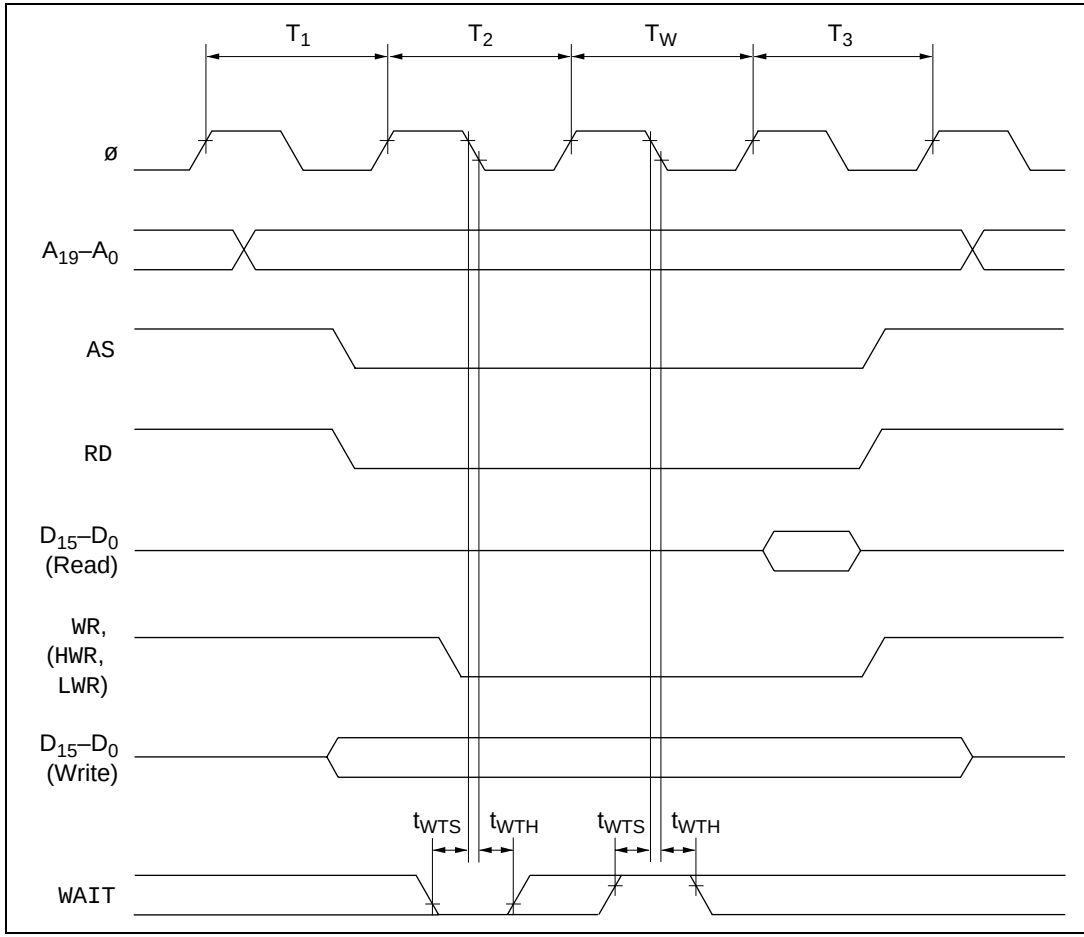


Figure 18.5 Basic Bus Cycle (With 1 Wait State) in Three-State Access Space

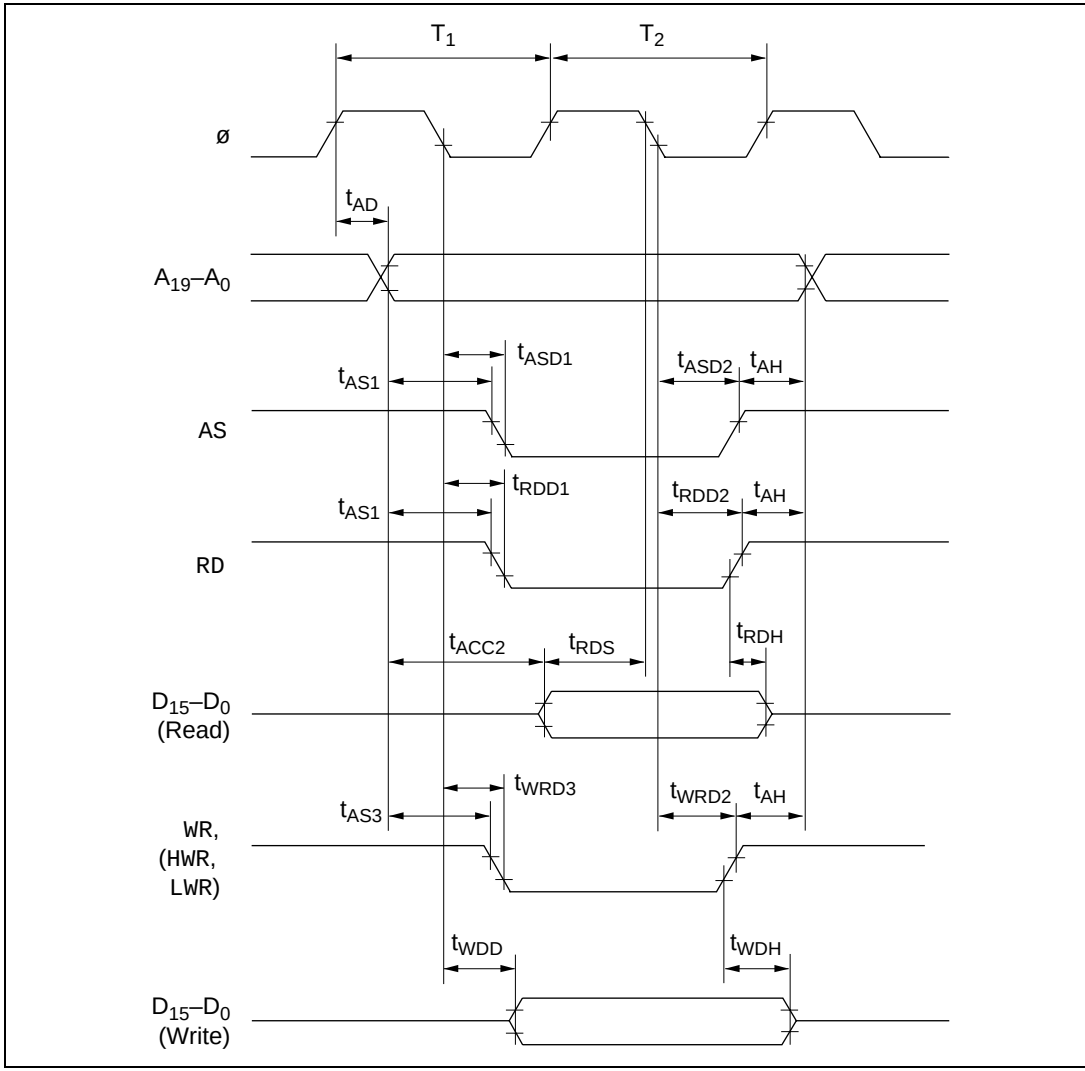


Figure 18.6 Basic Bus Cycle in Two-State Access Space

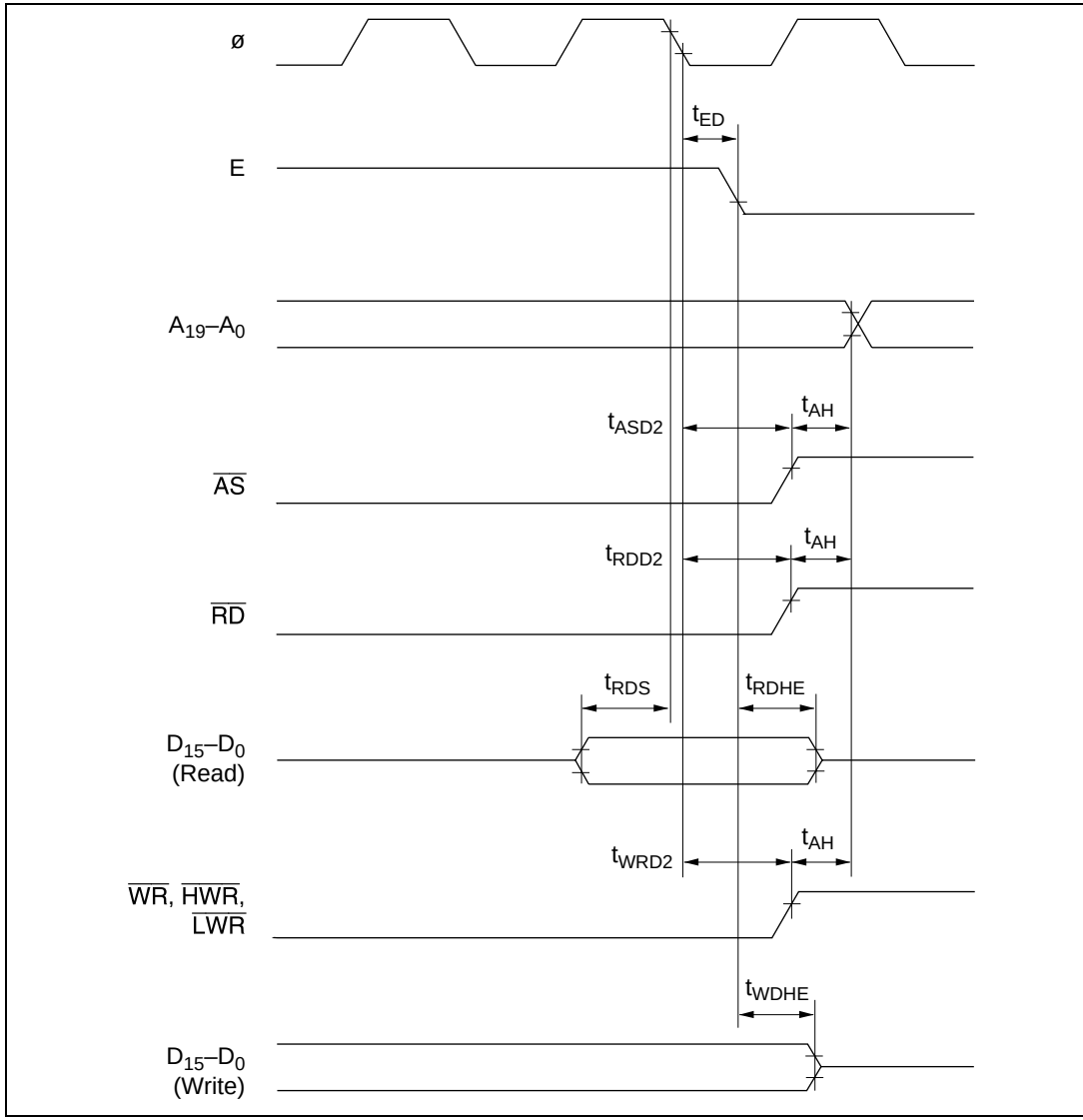


Figure 18.7 Bus Cycle Synchronized with E Clock

18.3.2 Control Signal Timing

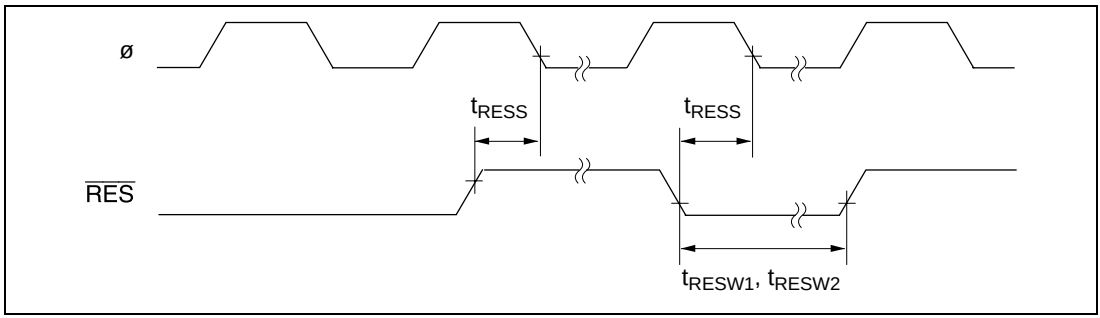


Figure 18.8 Reset Input Timing

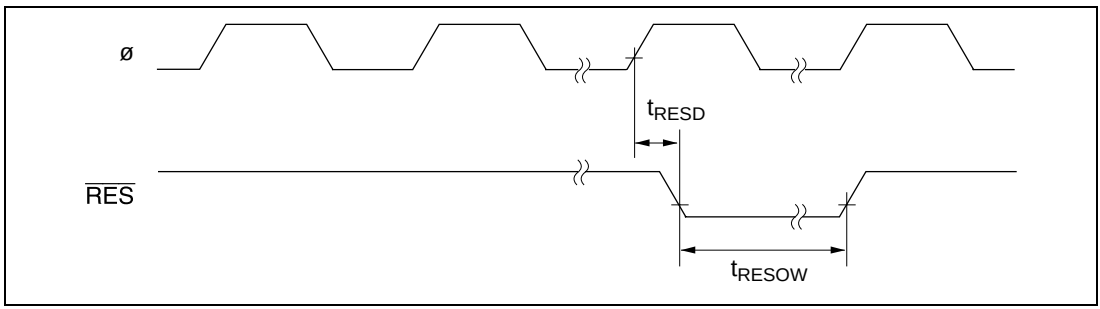


Figure 18.9 Reset Output Timing

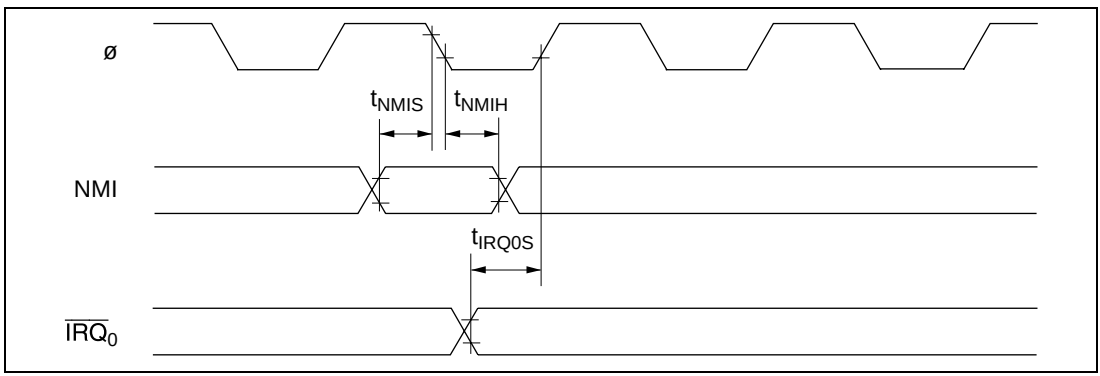


Figure 18.10 Interrupt Input Timing

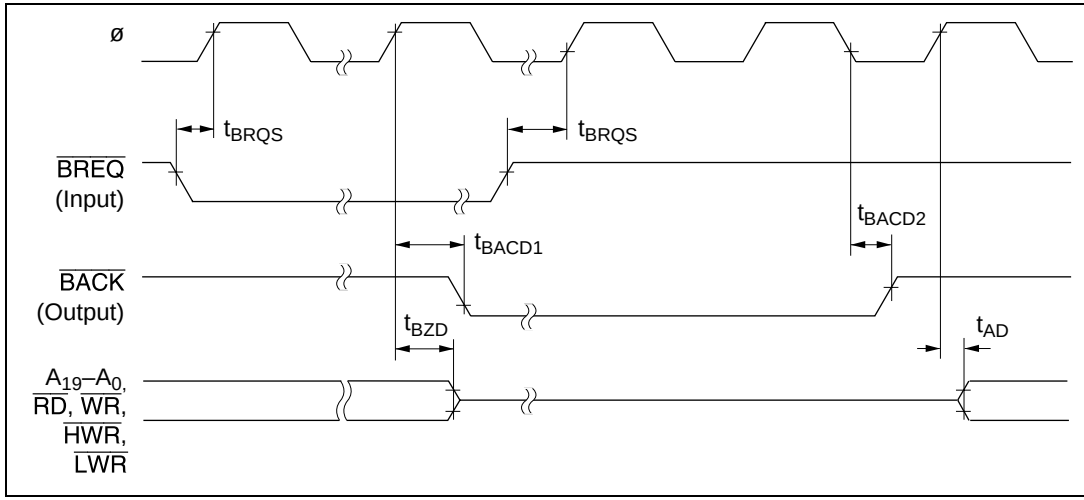


Figure 18.11 Bus Release State Timing

18.3.3 Clock Timing

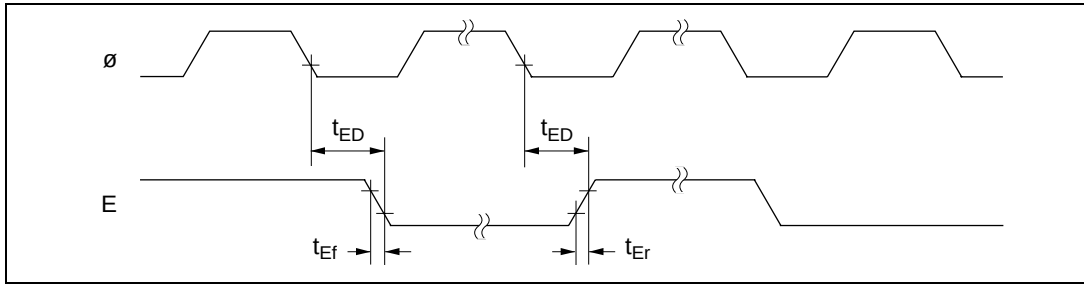


Figure 18.12 E-Clock Timing

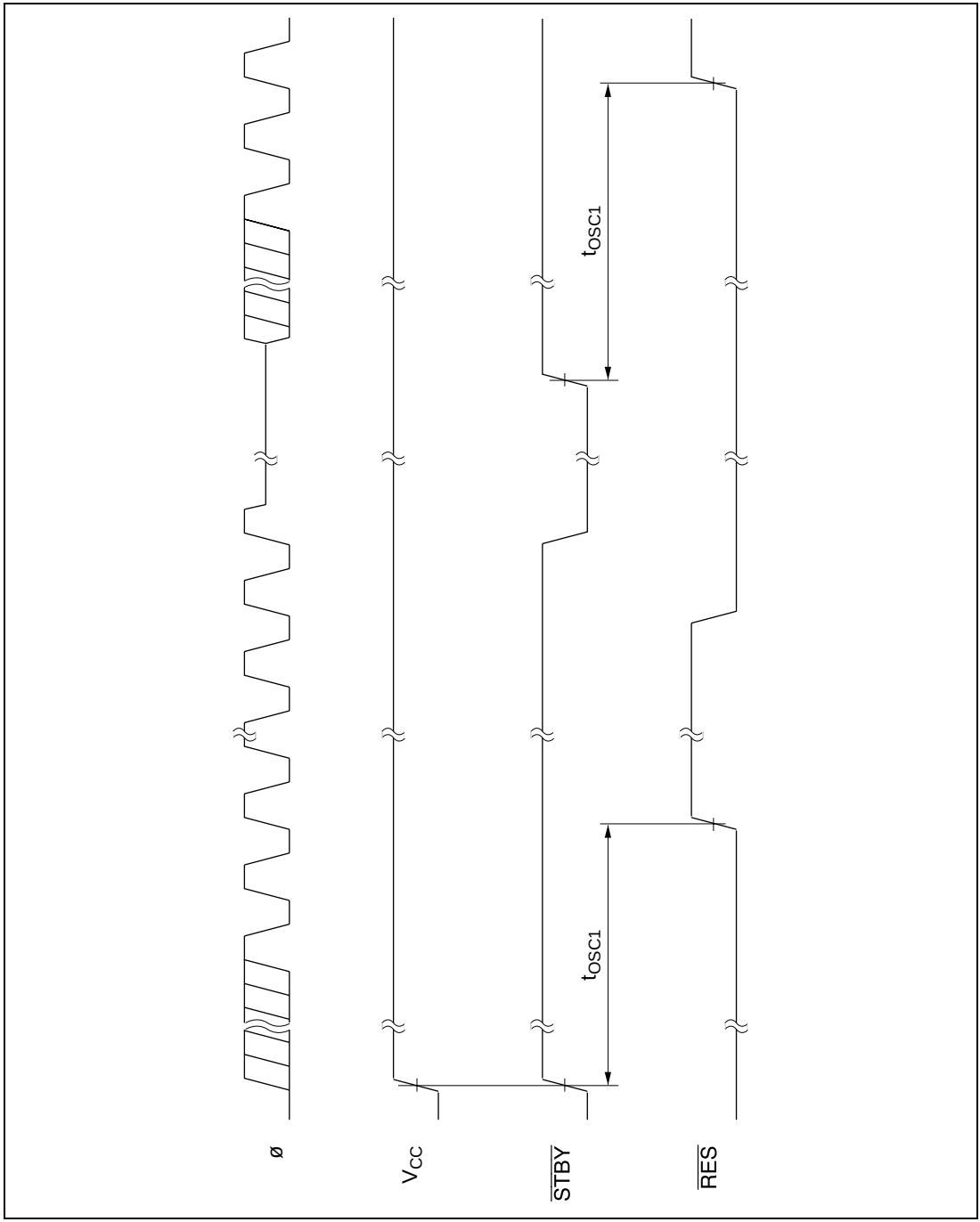


Figure 18.13 Clock Oscillator Stabilization Timing

18.3.4 I/O Port Timing

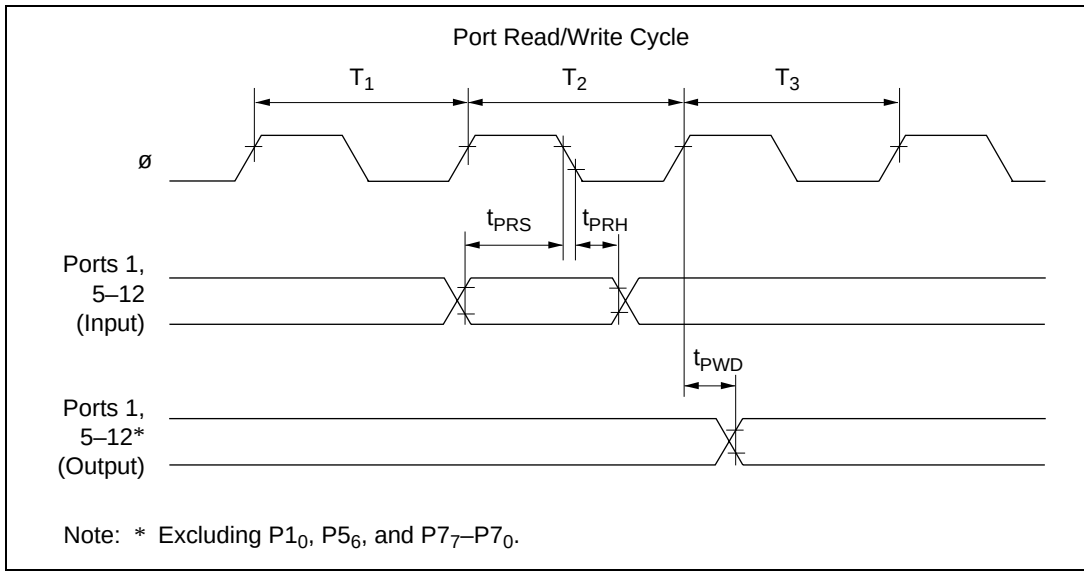


Figure 18.14 I/O Port Input/Output Timing

18.3.5 Pulse Width Modulation Timer Timing

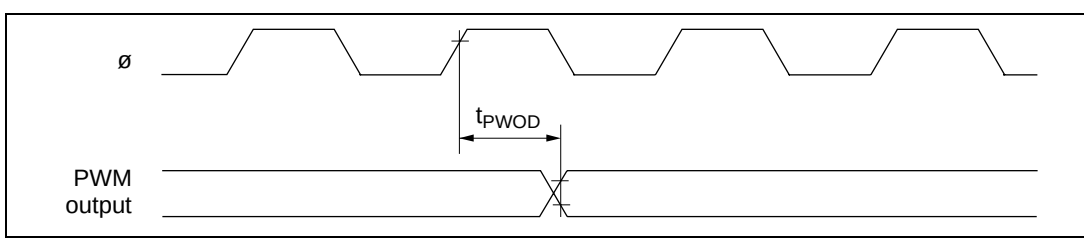


Figure 18.15 PWM Timer Output Timing

18.3.6 Serial Communication Interface Timing

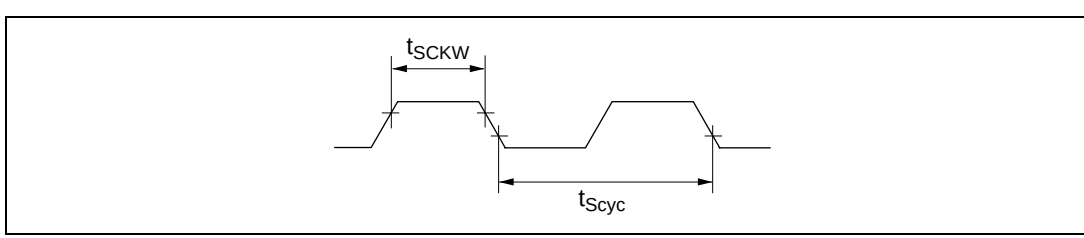


Figure 18.16 SCI Input Clock Timing

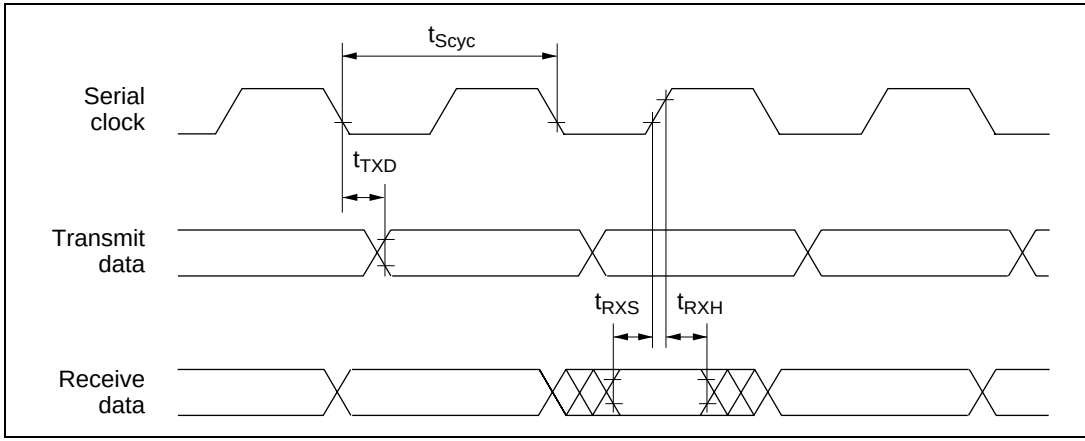


Figure 18.17 SCI Input/Output Timing (Synchronous Mode)

18.3.7 Intelligent Sub-Processor Interface Timing

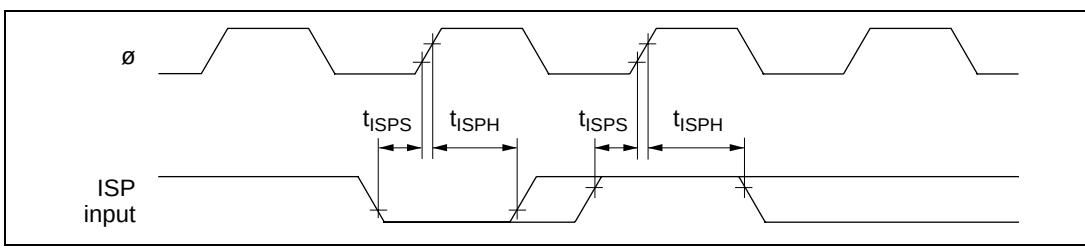


Figure 18.18 ISP Input Timing

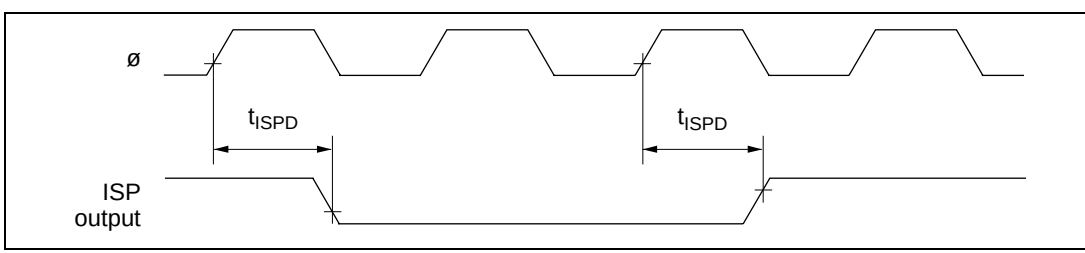


Figure 18.19 ISP Output Timing

Appendix A Instructions

A.1 Instruction Set

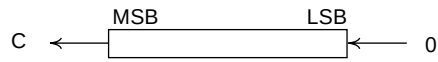
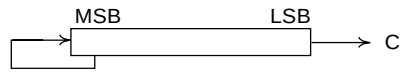
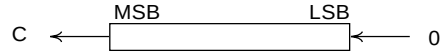
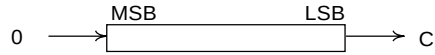
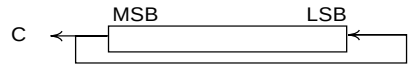
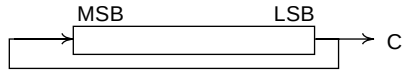
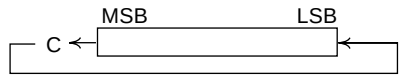
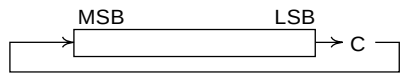
Operation Notation

Rd	General register (destination operand)
Rs	General register (source operand)
Rn	General register
(EAd)	Destination operand
(EAs)	Source operand
CCR	Condition code register
N	N (Negative) flag in CCR
Z	Z (Zero) flag in CCR
V	V (Overflow) flag in CCR
C	C (Carry) flag in CCR
CR	Control register
PC	Program counter
CP	Code page register
SP	Stack pointer
FP	Frame pointer
#Imm	Immediate data
disp	Displacement
+	Add
–	Subtract
×	Multiply
÷	Divide
∧	Logical AND
∨	Logical OR
⊕	Logical exclusive OR
→	Move
↔	Swap
¬	Logical NOT

Condition Code Notation

↕	Changed after instruction execution
0	Cleared to 0
1	Set to 1
—	Value before operation is retained
Δ	Conditional change

Mnemonic		Operation		Size	CCR Bit N	Z	V	C
Data transfer	MOV: G	(EAs) → Rd Rs → (EAd) #Imm → (EAd)		B/W	↕	↕	0	—
	MOV: E	#Imm → Rd	(short format)	B	↕	↕	0	—
	MOV: F	@(d:8, FP) →Rd Rs → @(d:8, FP)	(short format)	B/W	↕	↕	0	—
	MOV: I	#Imm → Rd	(short format)	W	↕	↕	0	—
	MOV: L	(@aa: 8) → Rd	(short format)	B/W	↕	↕	0	—
	MOV: S	Rs → (@aa: 8)	(short format)	B/W	↕	↕	0	—
	LDM	@SP+ → Rn (register list)		W	—	—	—	—
	STM	Rn (register list) → @–SP		W	—	—	—	—
	XCH	Rs ↔ Rd		W	—	—	—	—
	SWAP	Rd (upper byte) ↔ Rd (lower byte)		B	↕	↕	0	—
	MOVTPE	Rs → (EAd) Synchronized with E clock		B	—	—	—	—
	MOVFPE	(EAs) → Rd Synchronized with E clock		B	—	—	—	—
Arithmetic operations	ADD: G	Rd + (EAs) → Rd		B/W	↕	↕	↕	↕
	ADD: Q	(EAd) + #Imm → (EAd) (#Imm = ±1, ±2)	(short format)	B/W	↕	↕	↕	↕
	ADDS	Rd + (EAs) → Rd (Rd is always word size)		B/W	—	—	—	—
	ADDX	Rd + (EAs) + C → Rd		B/W	↕	↕	↕	↕
	DADD	$(Rd)_{10}$ + $(Rs)_{10}$ + C → $(Rd)_{10}$		B	—	↕	—	↕
	SUB	Rd – (EAs) → Rd		B/W	↕	↕	↕	↕
	SUBS	Rd – (EAs) → Rd		B/W	—	—	—	—
	SUBX	Rd – (EAs) – C → Rd		B/W	↕	↕	↕	↕
	DSUB	$(Rd)_{10}$ – $(Rs)_{10}$ – C → $(Rd)_{10}$		B	—	↕	—	↕
	MULXU	Rd × (EAs) → Rd (Unsigned)	8 × 8 16 × 16	B/W	↕	↕	0	0
	DIVXU	Rd ÷ (EAs) → Rd (Unsigned)	16 ÷ 8 32 ÷ 16	B/W	↕	↕	↕	0

Mnemonic		Operation	Size	CCR Bit N	Z	V	C
Arithmetic operations	CMP: G	Rd – (EAs), Set CCR (EAd) – #Imm, Set CCR	B/W	↕	↕	↕	↕
	CMP: E	Rd – #Imm, Set CCR (short format)	B	↕	↕	↕	↕
	CMP: I	Rd – #Imm, Set CCR (short format)	W	↕	↕	↕	↕
	EXTS	(<Bit 7> of <Rd>) → (<Bits 15 to 8> of <Rd>)	B	↕	↕	0	0
	EXTU	0 → (<Bits 15 to 8> of <Rd>)	B	0	↕	0	0
	TST	(EAd) – 0, Set CCR	B/W	↕	↕	0	0
	NEG	0 – (EAd) → (EAd)	B/W	↕	↕	0	↕
	CLR	0 → (EAd)	B/W	0	1	0	0
	TAS	(EAd) – 0, Set CCR 1 → (<Bit 7> of EAd>)	B	↕	↕	0	0
Shift operations	SHAL		B/W	↕	↕	↕	↕
	SHAR		B/W	↕	↕	0	↕
	SHLL		B/W	↕	↕	0	↕
	SHLR		B/W	0	↕	0	↕
	ROTL		B/W	↕	↕	0	↕
	ROTR		B/W	↕	↕	0	↕
	ROTXL		B/W	↕	↕	0	↕
	ROTXR		B/W	↕	↕	0	↕

Mnemonic		Operation	Size	CCR Bit N	Z	V	C
Logic operations	AND	Rd ∧ (EAs) → Rd	B/W	↕	↕	0	—
	OR	Rd ∨ (EAs) → Rd	B/W	↕	↕	0	—
	XOR	Rd ⊕ (EAs) → Rd	B/W	↕	↕	0	—
	NOT	¬ (EAd) → (EAd)	B/W	↕	↕	0	—
Bit manipulations	BSET	¬ (<Bit number> of <Rn>)→ Z 1 → (<Bit number> of <EAd>)	B/W	—	↕	—	—
	BCLR	¬ (<Bit number> of <Rn>)→ Z 0 → (<Bit number> of <EAd>)	B/W	—	↕	—	—
	BTST	¬ (<Bit number> of <EAd>)→ Z	B/W	—	↕	—	—
	BNOT	¬ (<Bit number> of <Rn>)→ Z → (<Bit number> of <EAd>)	B/W	—	↕	—	—
Branching instructions	Bcc	If condition is true then PC + disp → PC else next;	—	—	—	—	—

Mnemonic		Description	Condition
BRA	(BT)	Always (true)	True
BRN	(BF)	Never (false)	False
BHI		High	C ∨ Z = 0
BLS		Low or same	C ∨ Z = 0
BCC	(BHS)	Carry clear (high or same)	C = 0
BCS	(BLO)	Carry set (low)	C = 1
BNE		Not equal	Z = 0
BEQ		Equal	Z = 1
BVC		Overflow clear	V = 0
BVS		Overflow set	V = 1
BPL		Plus	N = 0
BMI		Minus	N = 1
BGE		Greater or equal	N ⊕ V = 0
BLT		Less than	N ⊕ V = 1
BGT		Greater than	Z ∨ (N ⊕ V) = 0
BLE		Less or equal	Z ∨ (N ⊕ V) = 1

Mnemonic		Operation	Size	CCR Bit N	Z	V	C
Branching instructions	JMP	Effective address → PC	—	—	—	—	—
	PJMP	Effective address → CP, PC	—	—	—	—	—
	BSR	PC → @–SP PC + disp → PC	—	—	—	—	—
	JSR	PC → @–SP Effective address → PC	—	—	—	—	—
	PJSR	PC → @–SP CP → @–SP Effective address → CP, PC	—	—	—	—	—
	RTS	@SP+ → PC	—	—	—	—	—
	PRTS	@SP+ → CP @SP+ → PC	—	—	—	—	—
	RTD	@SP+ → PC SP + #Imm → SP	—	—	—	—	—
	PRTD	@SP+ → CP @SP+ → PC SP + #Imm → SP	—	—	—	—	—
	SCB SCB/F SCB/NE SCB/EQ	If condition is true then next; else Rn – 1 → Rn; If Rn = –1 then next; else PC + disp → PC;	—	—	—	—	—

Mnemonic	Description	Condition
SCB/F		False
SCB/NE	Not equal	Z = 0
SCB/EQ	Equal	Z = 1

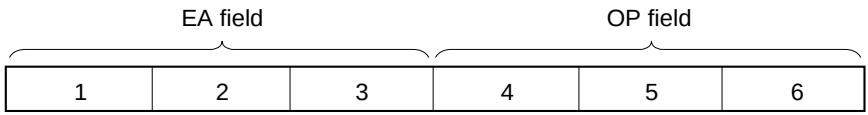
Mnemonic		Operation	Size	CCR Bit N	Z	V	C
System control	TRAPA	PC → @–SP (If max mode, CP → @–SP) SR → @–SP (If max mode, <vector> → CP) <vector> → PC	—	—	—	—	—
	TRAP/VS	If V bit = 1 then TRAP else next;	—	—	—	—	—
	RTE	@SP+ → SR (If max mode, @SP+ → CP) @SP+ → PC	—	↕	↕	↕	↕
	LINK	FP (R6) → @–SP SP → FP (R6) SP + #Imm → SP	—	—	—	—	—
	UNLK	FP (R6) → SP @SP+ → FP	—	—	—	—	—
	SLEEP	Normal running mode → Power-down state	—	—	—	—	—
	LDC	(EAs) → CR	B/W*	Δ	Δ	Δ	Δ
	STC	CR → (EAd)	B/W*	—	—	—	—
	ANDC	CR ∧ #Imm → CR	B/W*	Δ	Δ	Δ	Δ
	ORC	CR ∨ #Imm → CR	B/W*	Δ	Δ	Δ	Δ
	XORC	CR ⊕ #Imm → CR	B/W*	Δ	Δ	Δ	Δ
	NOP	PC + 1 → PC	—	—	—	—	—

Note: * Size depends on the CR.

A.2 Instruction Codes

How to Read Tables A.1 (a) to (d)

The general operand format consists of an effective address (EA) field and operation-code (OP) field specified in the following order:



Bytes 2, 3, 5, and 6 are not present in all instructions.

Instruction		Addressing Mode	Operation Code (EA) 1	2	3	Operation Code (OP) 4	5	6
uction	MOV:G.B <EAs>,Rd	Rn	1 0 1 0 Sz r r r			1 0 0 0 0 r_d r_d r_d		
		@Rn	1 1 0 1 Sz r r r					
		@(d:8, Rn)	1 1 1 0 Sz r r r	Displacement				
		@(d:16, Rn)	1 1 1 1 Sz r r r	Displacement (H)	Displacement (L)			
		@-Rn	1 0 1 1 Sz r r r					
		@Rn+	1 1 0 0 Sz r r r					
		@aa:8	0 0 0 0 Sz 1 0 1	Address				
		@aa:16	1 0 0 1 Sz 1 0 1	Address (H)	Address (L)			
		#xx:8	0 0 0 0 0 1 0 0	Data				
		#xx:16	0 0 0 0 1 1 0 0	Data (H)	Data (L)			
	MOV:G.W <EAs>,Rd	2	2	3	4	2	2	3	4		4	1 0 0 0 0 r_d r_d r_d		
	MOV:G.B Rs,<EAd>	2	2	3	4	2	2	3	4	3		1 0 0 1 0 r_s r_s r_s		
	MOV:G.B Rs,<EAd>	2	2	3	4	2	2	3	4		4	1 0 0 1 0 r_s r_s r_s		

Byte length of the instruction

Shading indicates the modes not available for the instruction.

Some instructions have a special format in which the operation code comes first.

Notations Used for Tables A.1 (a) to (d)

- Sz: Operand size (byte or word)
  Byte: Sz = 0
  Word: Sz = 1

- r r r: General register number field

r r r	Sz = 0 (Byte)		Sz = 1 (Word)
	15 8	7 0	15 0
0 0 0	Not used	R0	R0
0 0 1	Not used	R1	R1
0 1 0	Not used	R2	R2
0 1 1	Not used	R3	R3
1 0 0	Not used	R4	R4
1 0 1	Not used	R5	R5
1 1 0	Not used	R6	R6
1 1 1	Not used	R7	R7

- c c c: Control register number field

c c c	Sz = 0 (Byte)	Sz = 1 (Word)
		15 0
0 0 0	(Not allowed*)	SR
	7 0	
0 0 1	CCR	(Not allowed)
0 1 0	(Not allowed)	(Not allowed)
0 1 1	BR	(Not allowed)
1 0 0	EP	(Not allowed)
1 0 1	DP	(Not allowed)
1 1 0	(Not allowed)	(Not allowed)
1 1 1	TP	(Not allowed)

Note: * “Not allowed” values for ccc may cause abnormal results if specified.

- Register list: A byte with bits corresponding to the general registers as follows:

Bit:	7	6	5	4	3	2	1	0
	R7	R6	R5	R4	R3	R2	R1	R0

- #VEC: Four bits designating a vector number from 0 to 15. The vector numbers correspond to addresses of entries in the exception vector table as follows:

	Vector Address	
#VEC	**Minimum Mode**	**Maximum Mode**
0	H'0020–H'0021	H'0040–H'0043
1	H'0022–H'0023	H'0044–H'0047
2	H'0024–H'0025	H'0048–H'004B
3	H'0026–H'0027	H'004C–H'004F
4	H'0028–H'0029	H'0050–H'0053
5	H'002A–H'002B	H'0054–H'0057
6	H'002C–H'002D	H'0058–H'005B
7	H'002E–H'002F	H'005C–H'005F
8	H'0030–H'0031	H'0060–H'0063
9	H'0032–H'0033	H'0064–H'0067
10	H'0034–H'0035	H'0068–H'006B
11	H'0036–H'0037	H'006C–H'006F
12	H'0038–H'0039	H'0070–H'0073
13	H'003A–H'003B	H'0074–H'0077
14	H'003C–H'003D	H'0078–H'007B
15	H'003E–H'003F	H'007C–H'007F

Examples of machine-language coding

Example 1: `ADD:G.B @R0,R1`

	EA Field	OP Field
Table A.1(a)	1 1 0 1 Sz r r r	0 0 1 0 0 r_d r_d r_d
Machine code	1 1 0 1 0 0 0 0	0 0 1 0 0 0 0 1
	H'D021	

Example 2: `ADD:G.W @H'11:8,R1`

	EA Field		OP Field
Table A.1(a)	0 0 0 0 Sz 1 0 1	0 0 0 1 0 0 0 1	0 0 1 0 0 r_d r_d r_d
Machine code	0 0 0 0 1 1 0 1	0 0 0 1 0 0 0 1	0 0 1 0 0 0 0 1
	H'0D1121		

Table A.1 (a) Machine Language Coding (General Format)

	Instruction	Addressing Mode	Operation Code (EA) 1	Operation Code (EA) 2	Operation Code (EA) 3	Operation Code (OP) 4	Operation Code (OP) 5	Operation Code (OP) 6
Data transfer instruction	MOV:G.B <EAs>,Rd	Rn	1 0 1 0 Sz r r r			1 0 0 0 0 r_d r_d r_d		
		@Rn	1 1 0 1 Sz r r r					
		@(d:8, Rn)	1 1 1 0 Sz r r r	Displacement				
		@(d:16, Rn)	1 1 1 1 Sz r r r	Displacement (H)	Displacement (L)			
		@-Rn	1 0 1 1 Sz r r r					
		@Rn+	1 1 0 0 Sz r r r					
		@aa:8	0 0 0 0 Sz 1 0 1	Address				
		@aa:16	0 0 0 1 Sz 1 0 1	Address (H)	Address (L)			
		#xx:8	0 0 0 0 0 1 0 0	Data				
		#xx:16	0 0 0 0 1 1 0 0	Data (H)	Data (L)			
	MOV:G.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		1 0 0 0 0 r_d r_d r_d		
	MOV:G.W <EAs>,Rd	2	2	3	4	2	2	3	4		4	1 0 0 0 0 r_d r_d r_d		
	MOV:G.B Rs,<EAd>		2	3	4	2	2	3	4			1 0 0 1 0 r_s r_s r_s		
	MOV:G.W Rs,<EAd>		2	3	4	2	2	3	4			1 0 0 1 0 r_s r_s r_s		
	MOV:G.B #xx:8,<EAd>		3	4	5	3	3	4	5			0 0 0 0 0 1 1 0	Data	
	MOV:G.W #xx:8,<EAd>		3	4	5	3	3	4	5			0 0 0 0 0 1 1 0	Data	
	MOV:G.W #xx:16,<EAd>		4	5	6	4	4	5	6			0 0 0 0 0 1 1 1	Data (H)	Data (L)
	LDM.W @SP+,<register list>						2					0 0 0 0 0 0 1 0	Register list	
	STM.W <register list>,@-SP					2						0 0 0 1 0 0 1 0	Register list	
	XCH.W Rs,Rd	2										1 0 0 1 0 r_d r_d r_d		
	SWAP.B Rd	2										0 0 0 1 0 0 0 0		
	MOVTPE.B Rs,<EAd>		3	4	5	3	3	4	5			0 0 0 0 0 0 0 0	1 0 0 1 0 r_s r_s r_s	
	MOVTPE.B <EAs>,Rd		3	4	5	3	3	4	5			0 0 0 0 0 0 0 0	1 0 0 0 0 r_d r_d r_d	

	Instruction	Addressing Mode	Operation Code (EA) 1	2	3	Operation Code (OP) 4	5	6
Arithmetic operation instruction	ADD:G.B <EAs>,Rd	2	1 0 1 0 Sz r r r			0 0 0 1 0 r_d r_d r_d		
	ADD:G.W <EAs>,Rd	2	1 0 1 0 Sz r r r			0 0 0 1 0 r_d r_d r_d		
	ADD:Q.B #1,<EAd>*	2	1 1 0 1 Sz r r r			0 0 0 0 1 0 0 0		
	ADD:Q.W #1,<EAd>*	2	1 1 0 1 Sz r r r			0 0 0 0 1 0 0 0		
	ADD:Q.B #2,<EAd>*	2	1 1 1 0 Sz r r r	Displacement		0 0 0 0 1 0 0 1		
	ADD:Q.W #2,<EAd>*	2	1 1 1 0 Sz r r r	Displacement		0 0 0 0 1 0 0 1		
	ADD:Q.B #-1,<EAd>*	2	1 1 1 1 Sz r r r	Displacement (H)	Displacement (L)	0 0 0 0 1 1 0 0		
	ADD:Q.W #-1,<EAd>*	2	1 1 1 1 Sz r r r	Displacement (H)	Displacement (L)	0 0 0 0 1 1 0 0		
	ADD:Q.B #-2,<EAd>*	2	1 0 1 1 Sz r r r			0 0 0 0 1 1 0 1		
	ADD:Q.W #-2,<EAd>*	2	1 0 1 1 Sz r r r			0 0 0 0 1 1 0 1		
	ADDS.B <EAs>,Rd	2	1 1 0 0 Sz r r r			0 0 1 0 1 r_d r_d r_d		
	ADDS.W <EAs>,Rd	2	1 1 0 0 Sz r r r			0 0 1 0 1 r_d r_d r_d		
	ADDX.B <EAs>,Rd	2	0 0 0 0 Sz 1 0 1	Address		1 0 1 0 0 r_d r_d r_d		
	ADDX.W <EAs>,Rd	2	0 0 0 1 Sz 1 0 1	Address (H)	Address (L)	1 0 1 0 0 r_d r_d r_d		
		#xx:8	0 0 0 0 0 1 0 0	Data				
		#xx:16	0 0 0 0 1 1 0 0	Data (H)	Data (L)			

Note: * Short format instruction.

	Instruction	Addressing Mode: Rn	@Rn	@(d:8, Rn)	@(d:16, Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16	Operation Code (OP) 4	5	6
	Operation Code (EA) 1	1 0 1 0 Sz r r r	1 1 0 1 Sz r r r	1 1 1 0 Sz r r r	1 1 1 1 Sz r r r	1 0 1 1 Sz r r r	1 1 0 0 Sz r r r	0 0 0 0 Sz 1 0 1	0 0 0 1 Sz 1 0 1	0 0 0 0 0 1 0 0	0 0 0 0 1 1 0 0			
	Operation Code (EA) 2			Displacement	Displacement (H)			Address	Address (H)	Data	Data (H)			
	Operation Code (EA) 3				Displacement (L)				Address (L)		Data (L)			
Arithmetic operation instruction	DADD.B Rs,Rd	3										0 0 0 0 0 0 0 0	1 0 1 0 0 r_d r_d r_d	
	SUB.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		0 0 1 1 0 r_d r_d r_d		
	SUB.W <EAs>,Rd	2	2	3	4	2	2	3	4		4	0 0 1 1 0 r_d r_d r_d		
	SUBS.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		0 0 1 1 1 r_d r_d r_d		
	SUBS.W <EAs>,Rd	2	2	3	4	2	2	3	4		4	0 0 1 1 1 r_d r_d r_d		
	SUBX.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		1 0 1 1 0 r_d r_d r_d		
	SUBX.W <EAs>,Rd	2	2	3	4	2	2	3	4		4	1 0 1 1 0 r_d r_d r_d		
	DSUB.B Rs,Rd	3										0 0 0 0 0 0 0 0	1 0 1 1 0 r_d r_d r_d	
	MULXU.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		1 0 1 0 1 r_d r_d r_d		
	MULXU.W <EAs>,Rd	2	2	3	4	2	2	3	4		4	1 0 1 0 1 r_d r_d r_d		
	DIVXU.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		1 0 1 1 1 r_d r_d r_d		
	DIVXU.W <EAs>,Rd	2	2	3	4	2	2	3	4		4	1 0 1 1 1 r_d r_d r_d		
	CMP:G.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		0 1 1 1 0 r_d r_d r_d		
	CMP:G.W <EAs>,Rd	2	2	3	4	2	2	3	4		4	0 1 1 1 0 r_d r_d r_d		
	CMP:G.B #xx,<EAd>		3	4	5	3	3	4	5			0 0 0 0 0 1 0 0	Data	
	CMP:G.W #xx,<EAd>		4	5	6	4	4	5	6			0 0 0 0 0 1 0 1	Data (H)	Data (L)

	Instruction	Addressing Mode	Operation Code (EA) 1	2	3	Operation Code (OP) 4	5	6
		Rn	1 0 1 0 Sz r r r					
		@Rn	1 1 0 1 Sz r r r					
		@(d:8, Rn)	1 1 1 0 Sz r r r	Displacement				
		@(d:16, Rn)	1 1 1 1 Sz r r r	Displacement (H)	Displacement (L)			
		@-Rn	1 0 1 1 Sz r r r					
		@Rn+	1 1 0 0 Sz r r r					
		@aa:8	0 0 0 0 Sz 1 0 1	Address				
		@aa:16	0 0 0 1 Sz 1 0 1	Address (H)	Address (L)			
		#xx:8	0 0 0 0 0 1 0 0	Data				
		#xx:16	0 0 0 0 1 1 0 0	Data (H)	Data (L)			
Arithmetic operation instruction	EXTS.B Rd	2				0 0 0 1 0 0 0 1		
	EXTU.B Rd	2				0 0 0 1 0 0 1 0		
	TST.B <EAd>	2	2	3	4	0 0 0 1 0 1 1 0		
	TST.W <EAd>	2	2	3	4	0 0 0 1 0 1 1 0		
	NEG.B <EAd>	2	2	3	4	0 0 0 1 0 1 0 0		
	NEG.W <EAd>	2	2	3	4	0 0 0 1 0 1 0 0		
	CLR.B <EAd>	2	2	3	4	0 0 0 1 0 0 1 1		
	CLR.W <EAd>	2	2	3	4	0 0 0 1 0 0 1 1		
	TAS.B <EAd>	2	2	3	4	0 0 0 1 0 1 1 1		
Shift instruction	SHAL.B <EAd>	2	2	3	4	0 0 0 1 1 0 0 0		
	SHAL.W <EAd>	2	2	3	4	0 0 0 1 1 0 0 0		
	SHAR.B <EAd>	2	2	3	4	0 0 0 1 1 0 0 1		
	SHAR.W <EAd>	2	2	3	4	0 0 0 1 1 0 0 1		
	SHLL.B <EAd>	2	2	3	4	0 0 0 1 1 0 1 0		
	SHLL.W <EAd>	2	2	3	4	0 0 0 1 1 0 1 0		
	SHLR.B <EAd>	2	2	3	4	0 0 0 1 1 0 1 1		
	SHLR.W <EAd>	2	2	3	4	0 0 0 1 1 0 1 1		

	Instruction	Addressing Mode	Operation Code (EA) 1	Operation Code (EA) 2	Operation Code (EA) 3	Operation Code (OP) 4	Operation Code (OP) 5	Operation Code (OP) 6
		Rn	1 0 1 0 Sz r r r					
		@Rn	1 1 0 1 Sz r r r					
		@(d:8, Rn)	1 1 1 0 Sz r r r	Displacement				
		@(d:16, Rn)	1 1 1 1 Sz r r r	Displacement (H)	Displacement (L)			
		@-Rn	1 0 1 1 Sz r r r					
		@Rn+	1 1 0 0 Sz r r r					
		@aa:8	0 0 0 0 Sz 1 0 1	Address				
		@aa:16	0 0 0 1 Sz 1 0 1	Address (H)	Address (L)			
		#xx:8	0 0 0 0 0 1 0 0	Data				
		#xx:16	0 0 0 0 1 1 0 0	Data (H)	Data (L)			
Shift instruction	ROTL.B <EAd>	2	2	3	4	2	2	3	4			0 0 0 1 1 1 0 0		
	ROTL.W <EAd>	2	2	3	4	2	2	3	4			0 0 0 1 1 1 0 0		
	ROTR.B <EAd>	2	2	3	4	2	2	3	4			0 0 0 1 1 1 0 1		
	ROTR.W <EAd>	2	2	3	4	2	2	3	4			0 0 0 1 1 1 0 1		
	ROTXL.B <EAd>	2	2	3	4	2	2	3	4			0 0 0 1 1 1 1 0		
	ROTXL.W <EAd>	2	2	3	4	2	2	3	4			0 0 0 1 1 1 1 0		
	ROTXR.B <EAd>	2	2	3	4	2	2	3	4			0 0 0 1 1 1 1 1		
	ROTXR.W <EAd>	2	2	3	4	2	2	3	4			0 0 0 1 1 1 1 1		
Logic operation instruction	AND.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		0 1 0 1 0 r_d r_d r_d		
	AND.W <EAs>,Rd	2	2	3	4	2	2	3	4		4	0 1 0 1 0 r_d r_d r_d		
	OR.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		0 1 0 0 0 r_d r_d r_d		
	OR.W <EAs>,Rd	2	2	3	4	2	2	3	4		4	0 1 0 0 0 r_d r_d r_d		
	XOR.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		0 1 1 0 0 r_d r_d r_d		
	XOR.W <EAs>,Rd	2	2	3	4	2	2	3	4		4	0 1 1 0 0 r_d r_d r_d		
	NOT.B <EAd>	2	2	3	4	2	2	3	4			0 0 0 1 0 1 0 1		
	NOT.W <EAd>	2	2	3	4	2	2	3	4			0 0 0 1 0 1 0 1		

Instruction		Addressing Mode	Operation Code (EA) 1	2	3	Operation Code (OP) 4	5	6
Bit manipulation instruction	BSET.B #xx,<EAd>	Rn	1 0 1 0 Sz r r r			1 1 0 0 data		
	BSET.W #xx,<EAd>	Rn	1 0 1 0 Sz r r r			1 1 0 0 data		
	BSET.B Rs,<EAd>	Rn	1 1 0 1 Sz r r r			0 1 0 0 1 r_S r_S r_S		
	BSET.W Rs,<EAd>	Rn	1 1 0 1 Sz r r r			0 1 0 0 1 r_S r_S r_S		
	BCLR.B #xx,<EAd>	Rn	1 1 1 0 Sz r r r	Displacement		1 1 0 1 data		
	BCLR.W #xx,<EAd>	Rn	1 1 1 0 Sz r r r	Displacement		1 1 0 1 data		
	BCLR.B Rs,<EAd>	Rn	1 1 1 1 Sz r r r	Displacement (H)	Displacement (L)	0 1 0 1 1 r_S r_S r_S		
	BCLR.W Rs,<EAd>	Rn	1 1 1 1 Sz r r r	Displacement (H)	Displacement (L)	0 1 0 1 1 r_S r_S r_S		
	BTST.B #xx,<EAd>	Rn	1 0 1 1 Sz r r r			1 1 1 1 data		
	BTST.W #xx,<EAd>	Rn	1 0 1 1 Sz r r r			1 1 1 1 data		
	BTST.B Rs,<EAd>	Rn	1 1 0 0 Sz r r r			0 1 1 1 1 r_S r_S r_S		
	BTST.W Rs,<EAd>	Rn	1 1 0 0 Sz r r r			0 1 1 1 1 r_S r_S r_S		
	BNOT.B #xx,<EAd>	Rn	0 0 0 0 Sz 1 0 1	Address		1 1 1 0 data		
	BNOT.W #xx,<EAd>	Rn	0 0 0 0 Sz 1 0 1	Address		1 1 1 0 data		
	BNOT.B Rs,<EAd>	Rn	0 0 0 1 Sz 1 0 1	Address (H)	Address (L)	0 1 1 0 1 r_S r_S r_S		
	BNOT.W Rs,<EAd>	Rn	0 0 0 1 Sz 1 0 1	Address (H)	Address (L)	0 1 1 0 1 r_S r_S r_S		
		#xx:8	0 0 0 0 0 1 0 0	Data				
		#xx:16	0 0 0 0 1 1 0 0	Data (H)	Data (L)			

	Instruction	Addressing Mode	Operation Code (EA) 1	Operation Code (EA) 2	Operation Code (EA) 3	Operation Code (OP) 4	Operation Code (OP) 5	Operation Code (OP) 6
		Rn	1 0 1 0 Sz r r r			1 0 0 0 1 c c c		
		@Rn	1 1 0 1 Sz r r r			1 0 0 0 1 c c c		
		@(d:8, Rn)	1 1 1 0 Sz r r r	Displacement		1 0 0 1 1 c c c		
		@(d:16, Rn)	1 1 1 1 Sz r r r	Displacement (H)	Displacement (L)	1 0 0 1 1 c c c		
		@-Rn	1 0 1 1 Sz r r r			1 0 1 1 1 c c c		
		@Rn+	1 1 0 0 Sz r r r			1 0 1 1 1 c c c		
		@aa:8	0 0 0 0 Sz 1 0 1	Address		0 1 0 1 1 c c c		
		@aa:16	0 0 0 1 Sz 1 0 1	Address (H)	Address (L)	0 1 0 1 1 c c c		
		#xx:8	0 0 0 0 0 1 0 0	Data		0 1 0 0 1 c c c		
		#xx:16	0 0 0 0 1 1 0 0	Data (H)	Data (L)	0 1 0 0 1 c c c		
System control instruction	LDC.B <EAs>,CR	2	2	3	4	2	2	3	4	3		1 0 0 0 1 c c c		
	LDC.W <EAs>,CR	2	2	3	4	2	2	3	4		4	1 0 0 0 1 c c c		
	STC.B CR,<EAd>	2	2	3	4	2	2	3	4			1 0 0 1 1 c c c		
	STC.W CR,<EAd>	2	2	3	4	2	2	3	4			1 0 0 1 1 c c c		
	ANDC.B #xx:8,CR									3		0 1 0 1 1 c c c		
	ANDC.W #xx:16,CR										4	0 1 0 1 1 c c c		
	ORC.B #xx:8,CR									3		0 1 0 0 1 c c c		
	ORC.W #xx:16,CR										4	0 1 0 0 1 c c c		
	XORC.B #xx:8,CR									3		0 1 1 0 1 c c c		
	XORC.W #xx:16,CR										4	0 1 1 0 1 c c c		

Table A.1 (b) Machine Language Coding (Special Format: Short Version)

Instruction	Byte	Operation Code			
		1	2	3	4
MOV:E.B #xx:8,Rd	2	0 1 0 1 0 r_d r_d r_d	Data		
MOV:I.W #xx:16,Rd	3	0 1 0 1 1 r_d r_d r_d	Data	Data (L)	
MOV:L.B @aa:8,Rd	2	0 1 1 0 0 r_d r_d r_d	Address (L)		
MOV:L.W @aa:8,Rd	2	0 1 1 0 1 r_d r_d r_d	Address (L)		
MOV:S.B Rs,@aa:8	2	0 1 1 1 0 r_s r_s r_s	Address (L)		
MOV:S.W Rs,@aa:8	2	0 1 1 1 1 r_s r_s r_s	Address (L)		
MOV:F.B @(d:8,R6),Rd	2	1 0 0 0 0 r_d r_d r_d	Displacement		
MOV:F.W @(d:8,R6),Rd	2	1 0 0 0 1 r_d r_d r_d	Displacement		
MOV:F.B Rs,@(d:8,R6)	2	1 0 0 1 0 r_s r_s r_s	Displacement		
MOV:F.W Rs,@(d:8,R6)	2	1 0 0 1 1 r_s r_s r_s	Displacement		
CMP:E #xx:8,Rd	2	0 1 0 0 0 r_d r_d r_d	Data		
CMP:I #xx:16,Rd	3	0 1 0 0 1 r_d r_d r_d	Data (H)	Data (L)	

Table A.1 (c) Machine Language Coding (Special Format: Branching Instructions)

Instruction		Byte	Operation Code			
			1	2	3	4
Bcc d:8	BRA (BT)	2	0 0 1 0 0 0 0 0	Displacement		
	BRN (BF)		0 0 1 0 0 0 0 1	Displacement		
	BHI		0 0 1 0 0 0 1 0	Displacement		
	BLS		0 0 1 0 0 0 1 1	Displacement		
	BCC (BHS)		0 0 1 0 0 1 0 0	Displacement		
	BCS (BLO)		0 0 1 0 0 1 0 1	Displacement		
	BNE		0 0 1 0 0 1 1 0	Displacement		
	BEQ		0 0 1 0 0 1 1 1	Displacement		
	BVC		0 0 1 0 1 0 0 0	Displacement		
	BVS		0 0 1 0 1 0 0 1	Displacement		
	BPL		0 0 1 0 1 0 1 0	Displacement		
	BMI		0 0 1 0 1 0 1 1	Displacement		
	BGE		0 0 1 0 1 1 0 0	Displacement		
	BLT		0 0 1 0 1 1 0 1	Displacement		
	BGT		0 0 1 0 1 1 1 0	Displacement		
	BLE		0 0 1 0 1 1 1 1	Displacement		
Bcc d:16	BRA (BT)	3	0 0 1 1 0 0 0 0	Displacement (H)	Displacement (L)	
	BRN (BF)		0 0 1 1 0 0 0 1	Displacement (H)	Displacement (L)	
	BHI		0 0 1 1 0 0 1 0	Displacement (H)	Displacement (L)	
	BLS		0 0 1 1 0 0 1 1	Displacement (H)	Displacement (L)	
	BCC (BHS)		0 0 1 1 0 1 0 0	Displacement (H)	Displacement (L)	
	BCS (BLO)		0 0 1 1 0 1 0 1	Displacement (H)	Displacement (L)	
	BNE		0 0 1 1 0 1 1 0	Displacement (H)	Displacement (L)	
	BEQ		0 0 1 1 0 1 1 1	Displacement (H)	Displacement (L)	
	BVC		0 0 1 1 1 0 0 0	Displacement (H)	Displacement (L)	
	BVS		0 0 1 1 1 0 0 1	Displacement (H)	Displacement (L)	
	BPL		0 0 1 1 1 0 1 0	Displacement (H)	Displacement (L)	
	BMI		0 0 1 1 1 0 1 1	Displacement (H)	Displacement (L)	
	BGE		0 0 1 1 1 1 0 0	Displacement (H)	Displacement (L)	
	BLT		0 0 1 1 1 1 0 1	Displacement (H)	Displacement (L)	
	BGT		0 0 1 1 1 1 1 0	Displacement (H)	Displacement (L)	
	BLE		0 0 1 1 1 1 1 1	Displacement (H)	Displacement (L)	
JMP @Rn		2	0 0 0 1 0 0 0 1	1 1 0 1 0 r r r		
JMP @aa:16		3	0 0 0 1 0 0 0 0	Address (H)	Address (L)	

Instruction		Byte	Operation Code			
			1	2	3	4
JMP @(d:8,Rn)		3	0 0 0 1 0 0 0 1	1 1 1 0 0 r r r	Displacement	
JMP @(d:16,Rn)		4	0 0 0 1 0 0 0 1	1 1 1 1 0 r r r	Displacement (H)	Displacement (L)
BSR d:8		2	0 0 0 0 1 1 1 0	Displacement		
BSR d:16		3	0 0 0 1 1 1 1 0	Displacement (H)	Displacement (L)	
JSR @Rn		2	0 0 0 1 0 0 0 1	1 1 0 1 1 r r r		
JSR @aa:16		3	0 0 0 1 1 0 0 0	Address (H)	Address (L)	
JSR @(d:8,Rn)		3	0 0 0 1 0 0 0 1	1 1 1 0 1 r r r	Displacement	
JSR @(d:16,Rn)		4	0 0 0 1 0 0 0 1	1 1 1 1 1 r r r	Displacement (H)	Displacement (L)
RTS		1	0 0 0 1 1 0 0 1			
RTD #xx:8		2	0 0 0 1 0 1 0 0	Data		
RTD #xx:16		3	0 0 0 1 1 1 0 0	Data (H)	Data (L)	
SCB/cc Rn, disp	SCB/F	3	0 0 0 0 0 0 0 1	1 0 1 1 1 r r r	Displacement	
	SCB/NE		0 0 0 0 0 1 1 0	1 0 1 1 1 r r r	Displacement	
	SCB/EQ		0 0 0 0 0 1 1 1	1 0 1 1 1 r r r	Displacement	
PJMP @aa:24		4	0 0 0 1 0 0 1 1	Page	Address (H)	Address (L)
PJMP @Rn		2	0 0 0 1 0 0 0 1	1 1 0 0 0 r r r		
PJSR @aa:24		4	0 0 0 0 0 0 1 1	Page	Address(H)	Address (L)
PJSR @Rn		2	0 0 0 1 0 0 0 1	1 1 0 0 1 r r r		
PRTS		2	0 0 0 1 0 0 0 1	0 0 0 1 1 0 0 1		
PRTD #xx:8		3	0 0 0 1 0 0 0 1	0 0 0 1 0 1 0 0	Data	
PRTD #xx:16		4	0 0 0 1 0 0 0 1	0 0 0 1 1 1 0 0	Data (H)	Data (L)

Table A.1 (d) Machine Language Coding (Special Format: System Control Instructions)

Instruction	Byte	Operation Code			
		1	2	3	4
TRAPA #xx	2	0 0 0 0 1 0 0 0	0 0 0 1 #VEC		
TRAP/VS	1	0 0 0 0 1 0 0 1			
RTE	1	0 0 0 0 1 0 1 0			
LINK FP,#xx:8	2	0 0 0 1 0 1 1 1	Data		
LINK FP,#xx:16	3	0 0 0 1 1 1 1 1	Data (H)	Data (L)	
UNLK FP	1	0 0 0 0 1 1 1 1			
SLEEP	1	0 0 0 1 1 0 1 0			
NOP	1	0 0 0 0 0 0 0 0			

A.3 Operation Code Maps

Table A.2 Operation Codes in First Byte of Instruction

High \ Low	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	SCB/F See table A.6	LDM	PJSR @aa:24	#xx:8 See table A.5	#aa:8.B See table A.4	SCB/NE See table A.6	SCB/EQ See table A.6	TRAPA	TRAP/VS	RTE		#xx:16 See table A.5	@aa:8.W See table A.4	BSR d:8	UNLK
1	JMP	See table A.6*	STM	PJMP @aa:24	RTD #xx:8	@aa:16.B See table A.4		LINK #xx:8	JSR	RTS	SLEEP		RTD #xx:16	@aa:16.W See table A.4	BSR d:16	LINK #xx:16
2	BRA d:8	BRN	BHI	BLS	BCC	BCS	BNE	BEQ	BVC	BVS	BPL	BMI	BGE	BLT	BGT	BLE
3	BRA d:16	BRN	BHI	BLS	BCC	BCS	BNE	BEQ	BVC	BVS	BPL	BMI	BGE	BLT	BGT	BLE
4	R0	R1	CMP:E #xx:8,Rn R2 R3 R4			R5	R6	R7	R0	R1	CMP:I #xx:16,Rn R2 R3 R4			R5	R6	R7
5			MOV:E #xx:8,Rn								MOV:I #xx:16,Rn					
6			MOV:L.B @aa:8,Rn								MOV:L.W @aa:8,Rn					
7			MOV:S.B Rn,@aa:8								MOV:S.W Rn,@aa:8					
8			MOV:F.B @(d:8,R6),Rn								MOV:F.W @(d:8, R6),Rn					
9			MOV:F.B Rn,@(d:8,R6)								MOV:F.W Rn,@(d:8,R6)					
A			Rn	(Byte)	See table A.3						Rn	(Word)	See table A.3			
B			@–Rn	(Byte)	See table A.4						@–Rn	(Word)	See table A.4			
C			@Rn+	(Byte)	See table A.4						@Rn+	(Word)	See table A.4			
D			@Rn	(Byte)	See table A.4						@Rn	(Word)	See table A.4			
E			@(d:8,Rn)	(Byte)	See table A.4						@(d:8,Rn)	(Word)	See table A.4			
F			@(d:16,Rn)	(Byte)	See table A.4						@(d:16,Rn)	(Word)	See table A.4			

Notes: References to other tables indicate that additional bytes are included in the instruction code.
* H'11 is the first operation code byte of the following instructions:
JMP, JSR, PJMP, PJSR (register indirect addressing mode)
MP, JSR (register indirect addressing mode with displacement)
PRTS, PRTD (all addressing modes)

RENESAS

Table A.3 Operation Codes in Second Byte of Instruction (Axxx)

High \ Low	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	See table A.6*								ADD:Q #1	ADD:Q #2			ADD:Q #-1	ADD:Q #-2		
1	SWAP	EXTS	EXTU	CLR	NEG	NOT	TST	TAS	SHAL	SHAR	SHLL	SHLR	ROTL	ROTR	ROTXL	ROTXR
2	R0	R1	R2	R3	ADD R4	R5	R6	R7	R0	R1	R2	ADDS R3	R4	R5	R6	R7
3					SUB							SUBS				
4					OR						BSET (Register indirect specification of bit number)					
5					AND						BCLR (Register indirect specification of bit number)					
6					XOR						BNOT (Register indirect specification of bit number)					
7					CMP						BTST (Register indirect specification of bit number)					
8					MOV							LDC				
9					XCH							STC				
A					ADDX							MULXU				
B					SUBX							DIVXU				
C	b0	b1	b2	b3	b4	b5	BSET (Immediate specification of bit number) b6	b7	b8	b9	b10	b11	b12	b13	b14	b15
D							BCLR (Immediate specification of bit number)									
E							BNOT (Immediate specification of bit number)									
F							BTST (Immediate specification of bit number)									

Note: * The operation code is in the third byte.

Table A.4 Operation Codes in Second Byte of Instruction (05xx, 15xx, 0Dxx, 1Dxx, Bxxx, Cxxx, Dxxx, Exxx, Fxxx)

High \ Low	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	See table A.6*				CMP #xx:8	CMP #xx:16	MOV #xx:8	MOV #xx:16	ADD:Q #1	ADD:Q #2			ADD:Q #-1	ADD:Q #-2		
1				CLR	NEG	NOT	TST	TAS	SHAL	SHAR	SHLL	SHLR	ROTL	ROTR	ROTXL	ROTXR
2	R0	R1	R2	R3	ADD R4		R6	R7	R0	R1	R2	R3	ADDS R4	R5	R6	R7
3					SUB								SUBS			
4					OR						BSET (Register indirect specification of bit number)						
5					AND						BCLR (Register indirect specification of bit number)						
6					XOR						BNOT (Register indirect specification of bit number)						
7					CMP						BTST (Register indirect specification of bit number)						
8					MOV (load)								LDC			
9					MOV (store)								STC			
A					ADDX								MULXU			
B					SUBX								DIVXU			
C	b0	b1	b2	b3	b4	b5	BSET (Immediate specification of bit number) b6	b7	b8	b9	b10	b11	b12	b13	b14	b15
D							BCLR (Immediate specification of bit number)									
E							BNOT (Immediate specification of bit number)									
F							BTST (Immediate specification of bit number)									

Note: * The operation code is in the third byte.

RENESAS

Table A.5 Operation Codes in Second Byte of Instruction (04xx, 0Cxx)

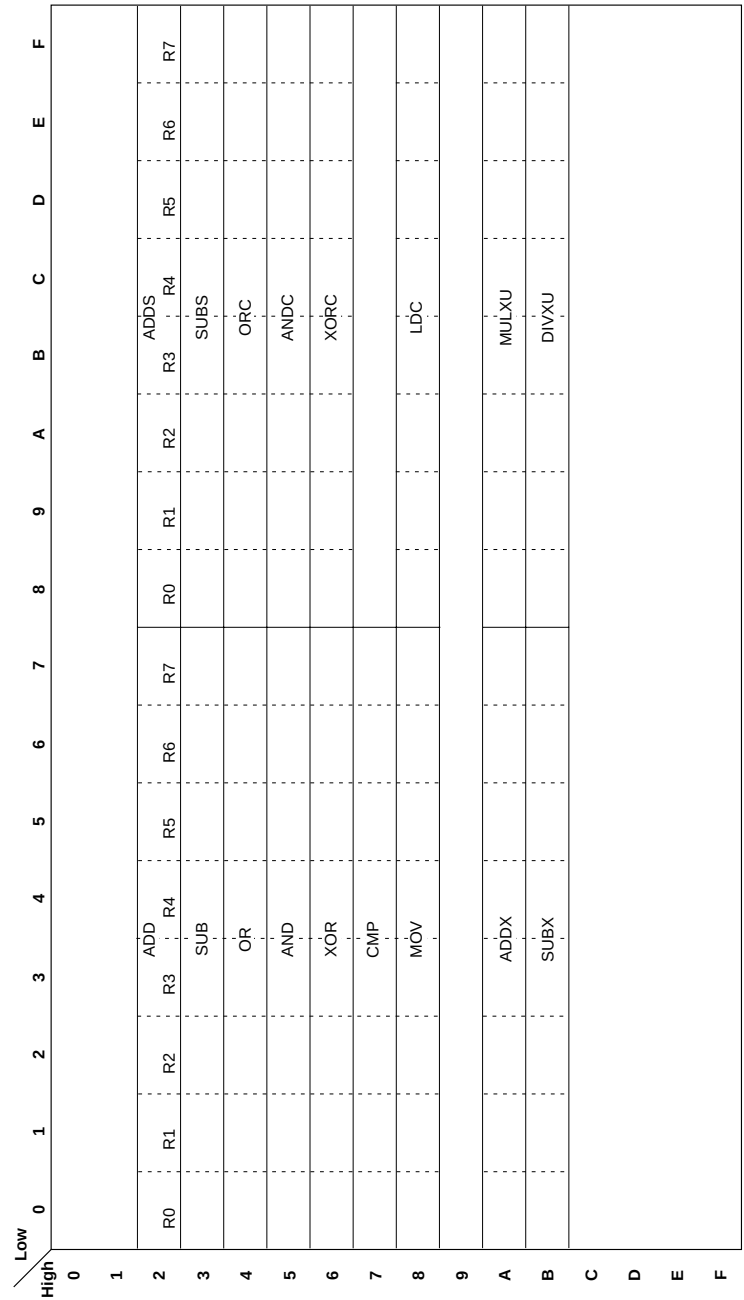
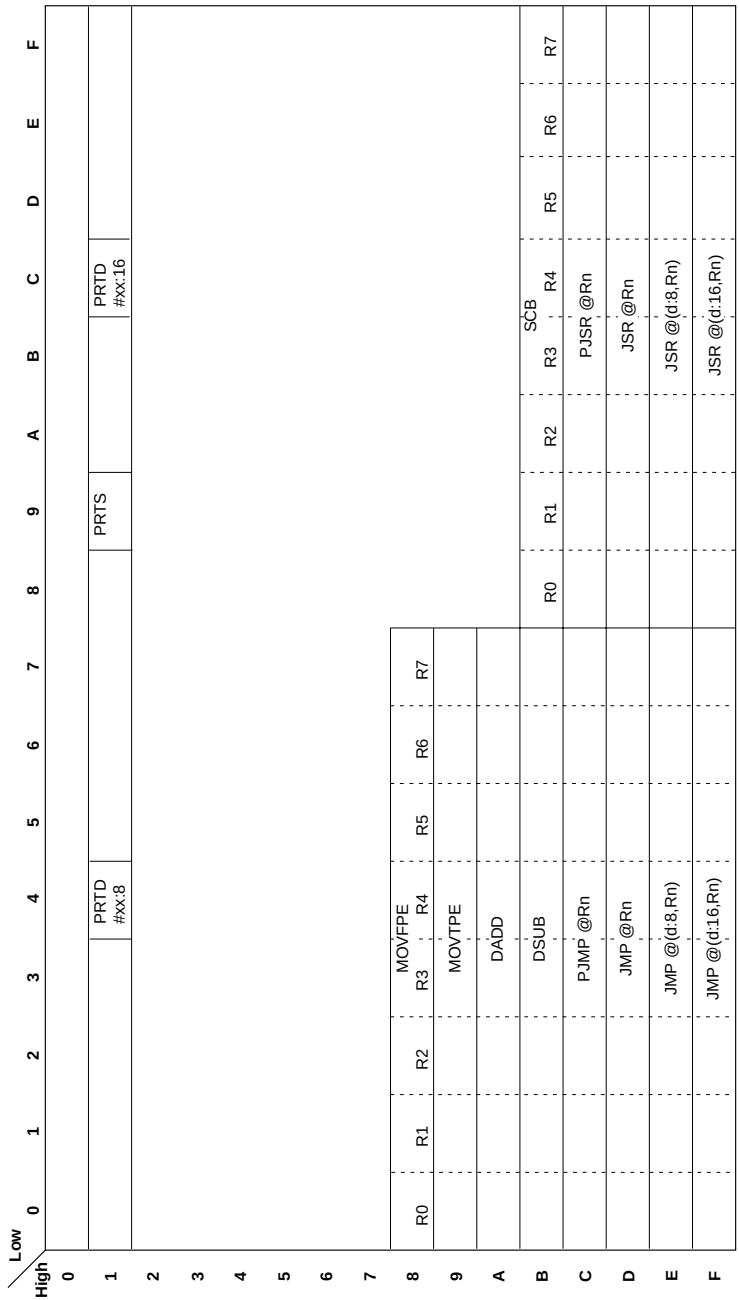


Table A.6 Operation Codes in Second and Third Bytes of Instruction (11xx, 01xx, 06xx, 07xx, xx00xx)



A.4 Instruction Execution Cycles

The values of I, J, and K are used to calculate the number of execution cycles needed to fetch an instruction or read/write an operand not located in the memory area and accessed in two states via a 16-bit bus.

Calculation of Instruction Execution States

One state is one cycle of the system clock (ø). If ø = 10 MHz, one state becomes 100 ns.

Instruction Fetch	Operand Read/Write	Number of States	
16-bit bus, two-state-access memory area	16-bit bus, two-state-access memory area or general register	(Value in table A.7) + (Value in table A.8)	
	16-bit bus, three-state-access memory area	Byte	(Value in table A.7) + (Value in table A.8) + I
		Word	(Value in table A.7) + (Value in table A.8) + I/2
	8-bit bus, three-state-access memory area or on-chip supporting module	Byte	(Value in table A.7) + (Value in table A.8) + I
		Word	(Value in table A.7) + (Value in table A.8) + 2 I
16-bit bus, three-state-access memory area	16-bit bus, two-state-access memory area or general register	(Value in table A.7) + (Value in table A.8) + (J + K) / 2	
	16-bit bus, three-state-access memory area	Byte	(Value in table A.7) + (Value in table A.8) + I + (J + K) / 2
		Word	(Value in table A.7) + (Value in table A.8) + (I + J + K) / 2
	8-bit bus, three-state-access memory area or on-chip supporting module	Byte	(Value in table A.7) + (Value in table A.8) + I + (J + K) / 2
		Word	(Value in table A.7) + (Value in table A.8) + 2 I + (J + K) / 2

Instruction Fetch	Operand Read/Write	Number of States	
8-bit bus, three-state-access memory area	16-bit bus, two-state-access memory area or general register	(Value in table A.7) + 2(J + K)	
	16-bit bus, three-state-access memory area	Byte	(Value in table A.7) + I + 2(J + K)
		Word	(Value in table A.7) + I/2 + 2(J + K)
	8-bit bus, three-state-access memory area or on-chip supporting module	Byte	(Value in table A.7) + I + 2(J + K)
		Word	(Value in table A.7) + 2(I + J + K)

Notes: 1. When the instruction is fetched via a 16-bit bus, the number of execution states varies by 1 or 2 depending on whether the instruction is stored at an even or odd address. This difference must be taken into consideration when the program is used for timing or when the exact number of states is required for a certain case.

2. If wait states or precharge states (T_P) are inserted during the access of the three-state-access memory area, the necessary number of states must be added.

3. When an instruction is fetched from a memory area that is accessed via a 16-bit bus in three states, the fractional result from the expression (J + K) / 2 should be rounded down.

Notes for Tables of Instruction Execution Cycles

J + K: Number of instruction fetch cycles.

I: Total number of bytes written and read when operand is in memory.

Instruction	I	J/K	Addressing Mode									
			Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@+Rn	@aa:8	@aa:16	#xx:8	#xx:16
			1	1	2	3	1	1	2	3	2	3
ADD:G.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3	
ADD:G.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6		4
ADD:Q.B #xx, <EAd>	2	1	2	7	7	8	7	8	7	8		
ADD:Q.W #xx, <EAd>	4	1	2	7	7	8	7	8	7	8		
ADDS.B <EAs>, Rd	1	1	3	5	5	6	5	6	5	6	3	
ADDS.W <EAs>, Rd	2	1	3	5	5	6	5	6	5	6		4
ADDX.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3	
ADDX.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6		4
AND.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3	
AND.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6		4
ANDC #xx, CR		1									5	9

Shading in the I column indicates that the operand cannot be in memory.

Shading in the addressing mode columns indicates the addressing modes that cannot be used with the instruction.

Examples of Calculating Number of States Required for Execution

Example 1: Instruction fetch from memory area accessed via 16-bit bus in 2 states

Operand Read/Write	Start Addr.	Assembler Notation			Table A.7 + Table A.8*	Number of States
		Address	Code	Mnemonic		
16-bit bus, two-state-access memory area or general register	Even	H'0100	H'D821	ADD @R0, R1	5 + 1	6
	Odd	H'0101	H'D821	ADD @R0, R1	5 + 0	5

Note: * Values referenced from tables A.7 and A.8.

Example 2: Instruction fetch from memory area accessed via 16-bit bus in 2 states

Operand Read/Write	Start Addr.	Assembler Notation			Table A.7 + Table A.8 + 2 I*	Number of States
		Address	Code	Mnemonic		
On-chip supporting module or 8-bit bus, three-state-access memory area (word)	Even	H'FC00	H'11D8	JSR @R0	9 + 0 + 2 × 2	13
	Odd	H'FC01	H'11D8	JSR @R0	9 + 1 + 2 × 2	14

Note: * Values referenced from tables A.7 and A.8.

Example 3: Instruction fetch from memory area accessed via 8-bit bus in 3 states

Operand Read/Write	Assembler Notation			Table A.7 + 2(J + K)*	Number of States
	Address	Code	Mnemonic		
16-bit bus, two-state-access memory area or general register	H'9002	H'D821	ADD @R0, R1	5 + 2(1 + 1)	9

Note: * Value referenced from table A.7.

Example 4: Instruction fetch from memory area accessed via 16-bit bus in 3 states

Operand Read/Write	Start Addr.	Assembler Notation			Table A.7 + Table A.8 (J + K) / 2*	Number of States
		Address	Code	Mnemonic		
16-bit bus, two-state-access memory area or general register	Even	H'0100	H'D821	ADD @R0, R1	5 + 1 + (1 + 1) / 2	7
	Odd	H'0101	H'D821	ADD @R0, R1	5 + 0 (1 + 1) / 2	6

Note: * Values referenced from tables A.7 and A.8.

Table A.7 Instruction Execution Cycles

			Addressing Mode									
			Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@+Rn	@aa:8	@aa:16	#xx:8	#xx:16
Instruction	**I**	**J/K**	1	1	2	3	1	1	2	3	2	3
ADD:G.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3	
ADD:G.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6		4
ADD:Q.B #xx, <EAd>	2	1	2	7	7	8	7	8	7	8		
ADD:Q.W #xx, <EAd>	4	1	2	7	7	8	7	8	7	8		
ADDS.B <EAs>, Rd	1	1	3	5	5	6	5	6	5	6	3	
ADDS.W <EAs>, Rd	2	1	3	5	5	6	5	6	5	6		4
ADDX.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3	
ADDX.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6		4
AND.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3	
AND.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6		4
ANDC #xx, CR		1									5	9
BCLR.B #xx, <EAd>*	2	1	4	7	7	8	7	8	7	8		
BCLR.W #xx, <EAd>*	4	1	4	7	7	8	7	8	7	8		
BNOT.B #xx, <EAd>*	2	1	4	7	7	8	7	8	7	8		
BNOT.W #xx, <EAd>*	4	1	4	7	7	8	7	8	7	8		
BSET.B #xx, <EAd>*	2	1	4	7	7	8	7	8	7	8		
BSET.W #xx, <EAd>*	4	1	4	7	7	8	7	8	7	8		
BTST.B #xx, <EAd>*	1	1	3	5	5	6	5	6	5	6		
BTST.W #xx, <EAd>*	2	1	3	5	5	6	5	6	5	6		
CLR.B <EAd>	1	1	2	5	5	6	5	6	5	6		
CLR.W <EAd>	2	1	2	5	5	6	5	6	5	6		
CMP:G.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3	
CMP:G.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6		4
CMP:G.B #xx:8, <EA>	1	2		6	6	7	6	7	6	7		
CMP:G.B #xx:16, <EA>	2	3		7	7	8	7	8	7	8		

Note: * Rs can also be used as the source operand.

			Addressing Mode									
			Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@+Rn	@aa:8	@aa:16	#xx:8	#xx:16
Instruction	**I**	**J/K**	1	1	2	3	1	1	2	3	2	3
CMP:E #xx:8, Rd		0									2	
CMP:I #xx:16, Rd		0										3
DADD Rs, Rd		2	4									
DIVXU.B <EAs>, Rd	1	1	20	23	23	24	23	24	23	24	21	
DIVXU.W <EAs>, Rd	2	1	26	29	29	30	29	30	29	30		28
DSUB Rs, Rd		2	4									
EXTS Rd		1	3									
EXTU Rd		1	3									
LDC.B <EAs>, CR	1	1	3	6	6	7	6	7	6	7	4	
LDC.W <EAs>, CR	2	1	4	7	7	8	7	8	7	8		6
MOV:G.B	1	1	2	5	5	6	5	6	5	6	3	
MOV:G.W	2	1	2	5	5	6	5	6	5	6		4
MOV.G.B #xx:8, <EAd>	1	2		7	7	8	7	8	7	8		
MOV.G.W #xx:16, <EAd>	2	3		8	8	9	8	9	8	9		
MOV:E #xx:8, Rd		0									2	
MOV:I #xx:16, Rd		0										3
MOV:L.B @aa:8, Rd	1	0							5			
MOV:L.W @aa:8, Rd	2	0							5			
MOV:S.B Rs, @aa:8	1	0							5			
MOV:S.W Rs, @aa:8	2	0							5			
MOV:F.B @(d:8, R6), Rd	1	0			5							
MOV:F.W @(d:8, R6), Rd	2	0			5							
MOV:F.B Rs, @(d:8, R6)	1	0			5							
MOV:F.W Rs, @(d:8, R6)	2	0			5							

			Addressing Mode									
			Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@+Rn	@aa:8	@aa:16	#xx:8	#xx:16
Instruction	**I**	**J/K**	1	1	2	3	1	1	2	3	2	3
MOVFPE* <EAs>, Rd	0	2		13↔20	13↔20	14↔21	13↔20	14↔21	13↔20	14↔21		
MOVTPE* Rd, <EAs>	0	2		13↔20	13↔20	14↔21	13↔20	14↔21	13↔20	14↔21		
MULXU.B <EAs>, Rd	1	1	16	19	19	20	19	20	19	20	18	
MULXU.W <EAs>, Rd	2	1	23	25	25	26	25	26	25	26		25
NEG.B <EAd>	2	1	2	7	7	8	7	8	7	8		
NEG.W <EAd>	4	1	2	7	7	8	7	8	7	8		
NOT.B <EAd>	2	1	2	7	7	8	7	8	7	8		
NOT.W <EAd>	4	1	2	7	7	8	7	8	7	8		
OR.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3	
OR.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6		4
ORC #xx, CR		1									5	9
ROTL.B <EAd>	2	1	2	7	7	8	7	8	7	8		
ROTL.W <EAd>	4	1	2	7	7	8	7	8	7	8		
ROTR.B <EAd>	2	1	2	7	7	8	7	8	7	8		
ROTR.W <EAd>	4	1	2	7	7	8	7	8	7	8		
ROTXL.B <EAd>	2	1	2	7	7	8	7	8	7	8		
ROTXL.W <EAd>	4	1	2	7	7	8	7	8	7	8		
ROTXR.B <EAd>	2	1	2	7	7	8	7	8	7	8		
ROTXR.W <EAd>	4	1	2	7	7	8	7	8	7	8		
SHAL.B <EAd>	2	1	2	7	7	8	7	8	7	8		
SHAL.W <EAd>	4	1	2	7	7	8	7	8	7	8		
SHAR.B <EAd>	2	1	2	7	7	8	7	8	7	8		
SHAR.W <EAd>	4	1	2	7	7	8	7	8	7	8		
SHLL.B <EAd>	2	1	2	7	7	8	7	8	7	8		
SHLL.W <EAd>	4	1	2	7	7	8	7	8	7	8		

Note: * Synchronous execution with the E clock so the number of execution states varies depending on the timing of the execution.

Instruction	I	J/K	Addressing Mode									
			Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@+Rn	@aa:8	@aa:16	#xx:8	#xx:16
			1	1	2	3	1	1	2	3	2	3
SHLR.B <EAd>	2	1	2	7	7	8	7	8	7	8		
SHLR.W <EAd>	4	1	2	7	7	8	7	8	7	8		
STC.B CR, <EAd>	1	1	4	7	7	8	7	8	7	8		
STC.W CR, <EAd>	2	1	4	7	7	8	7	8	7	8		
SUB.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3	
SUB.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6		4
SUBS.B <EAs>, Rd	1	1	3	5	5	6	5	6	5	6	3	
SUBS.W <EAs>, Rd	2	1	3	5	5	6	5	6	5	6		4
SUBX.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3	
SUBX.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6		4
SWAP Rd		1	3									
TAS <EAd>	2	1	4	7	7	8	7	8	7	8		
TST.B <EAd>	1	1	2	5	5	6	5	6	5	6		
TST.W <EAd>	2	1	2	5	5	6	5	6	5	6		
XCH Rs, Rd		1	4									
XOR.B <EAs>, Rd	1	1	2	5	6	5	5	6	5	6	3	
XOR.W <EAs>, Rd	2	1	2	5	6	5	5	6	5	6		4
XORC #xx, CR		1									5	9
DIVXU.B Zero divide, minimum mode	6/7*	1	20	23	23	24	23	24	23	24	21	
DIVXU.B Zero divide, maximum mode	10/11	1	25	28	28	29	28	29	28	29	21	
DIVXU.W Zero divide, minimum mode	6/8	1	20	23	23	24	23	24	23	24		27
DIVXU.W Zero divide, maximum mode	10/12	1	25	28	28	29	28	29	28	29		27
DIVXU.B Overflow	1	1	8	11	11	12	11	12	11	12	9	
DIVXU.W Overflow	2	1	8	11	11	12	11	12	11	12		10

Note: * The two state values of I under one instruction are for register and immediate operands (left value) and for memory operands (right value).

Instruction	Condition	Execution Cycles	I	J + K
Bcc d:8	Condition false, branch not taken	3	—	2
	Condition true, branch taken	7	—	5
Bcc d:16	Condition false, branch not taken	3	—	3
	Condition true, branch taken	7	—	6
BSR	d:8	9	2	4
	d:16	9	2	5
JMP	@aa:16	7	—	5
	@Rn	6	—	5
	@(d:8, Rn)	7	—	5
	@(d:16, Rn)	8	—	6
JSR	@aa:16	9	2	5
	@Rn	9	2	5
	@(d:8, Rn)	9	2	5
	@(d:16, Rn)	10	2	6
LDM		6 + 4n*	2n	2
LINK	#xx:8	6	2	2
	#xx:16	7	2	3
NOP		2	—	1
RTD	#xx:8	9	2	4
	#xx:16	9	2	5
RTE	Minimum mode	13	4	4
	Maximum mode	15	6	4
RTS		8	2	4
SCB	Condition false, branch not taken	3	—	3
	Count = –1, branch not taken	4	—	3
	Other than the above, branch taken	8	—	6
SLEEP	Cycles preceding transition to power-down mode	2	—	0
STM		6 + 3n	2n	2
TRAPA	Minimum mode	17	6	4
	Maximum mode	22	10	4
TRAP/VS	V = 0, trap not taken	3	—	1
	V = 1, trap taken, minimum mode	18	6	4
	V = 1, trap taken, maximum mode	23	10	4

Instruction	Condition	Execution Cycles	I	J + K
UNLK		5	2	1
PJMP	@aa:24	9	—	6
	@Rn	8	—	5
PJSR	@aa:24	15	4	6
	@Rn	13	4	5
PRTS		12	4	5
PRTD	#xx:8	13	4	5
	#xx:16	13	4	6

Note: * n is the number of registers specified in the register list.

Table A.8 (a) Adjustment Value (Branch Instructions)

Instruction	Address	Adjustment Value
BSR, JMP, JSR, RTS, RTD, RTE, TRAPA, PJMP, PJSR, PRTS, PRTD	Even	0
	Odd	1
Bcc, SCB, TRAP/VS (branch taken)	Even	0
	Odd	1

Table A.8 (b) Adjustment Value (Other Instructions by Addressing Modes)

Instruction	Start Address	Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@+Rn	@aa:8	@aa:16	#xx:8	#xx:16
MOV.B #xx:8, <EA>	Even		1	1	1	1	1	1	1		
MOVTPE, MOVFPE	Odd		1	1	1	1	1	1	1		
MOV.W #xx:16, <EA>	Even		2	0	2	2	2	0	2		
	Odd		0	2	0	0	0	2	0		
Instructions other than above	Even	0	1	0	1	1	1	0	1	0	0
	Odd	0	0	1	0	0	0	1	0	0	0

Appendix B Register Field

B.1 Register Addresses and Bit Names

Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FE80	ADDRAH	AD_9	AD_8	AD_7	AD_6	AD_5	AD_4	AD_3	AD_2	A/D
H'FE81	ADDRAL	AD_1	AD_0	—	—	—	—	—	—	
H'FE82	ADDRBH	AD_9	AD_8	AD_7	AD_6	AD_5	AD_4	AD_3	AD_2	
H'FE83	ADDRBL	AD_1	AD_0	—	—	—	—	—	—	
H'FE84	ADDRCH	AD_9	AD_8	AD_7	AD_6	AD_5	AD_4	AD_3	AD_2	
H'FE85	ADDRCL	AD_1	AD_0	—	—	—	—	—	—	
H'FE86	ADDRDH	AD_9	AD_8	AD_7	AD_6	AD_5	AD_4	AD_3	AD_2	
H'FE87	ADDRDL	AD_1	AD_0	—	—	—	—	—	—	
H'FE88	ADCSR	ADF	ADIE	ADST	SCAN	CKS	CH2	CH1	CH0	
H'FE89	ADCR	TRGE	—	—	—	—	—	—	—	
H'FE8A	Write controller/ TCSR	OVF	WT/$\overline{IT}$	TME	—	—	CKS2	CKS1	CKS0	WDT
H'FE8B	TCSR/ TCNT									
H'FE8C	P1DR	$P1_7$	—	—	—	$P1_3$	$P1_2$	$P1_1$	$P1_0$	Port 1
H'FE8D	—	—	—	—	—	—	—	—	—	—
H'FE8E	—	—	—	—	—	—	—	—	—	
H'FE8F	—	—	—	—	—	—	—	—	—	
H'FE90	P5DR	$P5_7$	$P5_6$	—	$P5_4$	$P5_3$	$P5_2$	$P5_1$	$P5_0$	Port 5
H'FE91	P6DR	—	—	$P6_5$	$P6_4$	$P6_3$	$P6_2$	$P6_1$	$P6_0$	Port 6
H'FE92	P7DR	$P7_7$	$P7_6$	$P7_5$	$P7_4$	$P7_3$	$P7_2$	$P7_1$	$P7_0$	Port 7
H'FE93	P8DR	$P8_7$	$P8_6$	$P8_5$	$P8_4$	$P8_3$	$P8_2$	$P8_1$	$P8_0$	Port 8
H'FE94	P9DR	$P9_7$	$P9_6$	$P9_5$	$P9_4$	$P9_3$	$P9_2$	$P9_1$	$P9_0$	Port 9
H'FE95	P10DR	$P10_7$	$P10_6$	$P10_5$	$P10_4$	$P10_3$	$P10_2$	$P10_1$	$P10_0$	Port 10
H'FE96	P11DR	$P11_7$	$P11_6$	$P11_5$	$P11_4$	$P11_3$	$P11_2$	$P11_1$	$P11_0$	Port 11
H'FE97	P12DR	$P12_7$	$P12_6$	$P12_5$	$P12_4$	$P12_3$	$P12_2$	$P12_1$	$P12_0$	Port 12

Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FE98	SMR	C/$\overline{A}$	CHR	PE	O/$\overline{E}$	STOP	—	CKS1	CKS0	SCI
H'FE99	BRR									
H'FE9A	SCR	TIE	RIE	TE	RE	—	—	CKE1	CKE0	
H'FE9B	TDR									
H'FE9C	SSR	TDRE	RDRF	ORER	FER	PER	—	—	—	
H'FE9D	RDR									
H'FE9E	—	—	—	—	—	—	—	—	—	
H'FE9F	—	—	—	—	—	—	—	—	—	
H'FEA0	TCR	TCE	$\overline{FRM}$	CKS1	CKS0	OMS	OE2	OE1	OE0	PWM
H'FEA1	TSR	TRE2	TRE0	OCIE2	OCIE1	OCIE0	OCF2	OCF1	OCF0	
H'FEA2	ODL	—	—	ODL5	ODL4	ODL3	ODL2	ODL1	ODL0	
H'FEA3	ODR0	—	—	OD05	OD04	OD03	OD02	OD01	OD00	
H'FEA4	ODR1	—	—	OD15	OD14	OD13	OD12	OD11	OD10	
H'FEA5	ODR2	—	—	OD25	OD24	OD23	OD22	OD21	OD20	
H'FEA6	OCR0H									
H'FEA7	OCR0L									
H'FEA8	OCR1H									
H'FEA9	OCR1L									
H'FEAA	OCR2H									
H'FEAB	OCR2L									
H'FEAC	TMRH									
H'FEAD	TMRL									
H'FEAE	—	—	—	—	—	—	—	—	—	—
H'FEAF	—	—	—	—	—	—	—	—	—	
H'FEB0	ISFH	ISFH7	ISFH6	ISFH5	ISFH4	ISFH3	ISFH2	ISFH1	ISFH0	ISP
H'FEB1	ISFL	ISFL7	ISFL6	ISFL5	ISFL4	ISFL3	ISFL2	ISFL1	ISFL0	
H'FEB2	IOF2	IOF27	IOF26	IOF25	IOF24	IOF23	IOF22	IOF21	IOF20	
H'FEB3	IOF1	IOF17	IOF16	IOF15	IOF14	IOF13	IOF12	IOF11	IOF10	
H'FEB4	IOF0	IOF07	IOF06	IOF05	IOF04	IOF03	IOF02	IOF01	IOF00	
H'FEB5	EGF	EGF7	EGF6	EGF5	EGF4	EGF3	EGF2	EGF1	EGF0	

Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEB6	ICFH	ICFH7	ICFH6	ICFH5	ICFH4	ICFH3	ICFH2	ICFH1	ICFH0	ISP
H'FEB7	ICFL	ICFL7	ICFL6	ICFL5	ICFL4	ICFL3	ICFL2	ICFL1	ICFL0	
H'FEB8	IEFH	IEFH7	IEFH6	IEFH5	IEFH4	IEFH3	IEFH2	IEFH1	IEFH0	
H'FEB9	IEFL	IEFL7	IEFL6	IEFL5	IEFL4	IEFL3	IEFL2	IEFL1	IEFL0	
H'FEBA	IOIEH	IOIEH7	IOIEH6	IOIEH5	IOIEH4	IOIEH3	IOIEH2	IOIEH1	IOIEH0	
H'FEBB	IOIEL	IOIEL7	IOIEL6	IOIEL5	IOIEL4	IOIEL3	IOIEL2	IOIEL1	IOIEL0	
H'FEBC	CLEH	CLEH7	CLEH6	CLEH5	CLEH4	CLEH3	CLEH2	CLEH1	CLEH0	
H'FEBD	CLEL	CLEL7	CLEL6	CLEL5	CLEL4	CLEL3	CLEL2	CLEL1	CLEL0	
H'FEBE	—	—	—	—	—	—	—	—	—	
H'FEBF	EVER	—	—	—	—	—	EVE2	EVE1	EVE0	
H'FEC0	DR0H									
H'FEC1	DR0L									
H'FEC2	DR1H									
H'FEC3	DR1L									
H'FEC4	DR2H									
H'FEC5	DR2L									
H'FEC6	DR3H									
H'FEC7	DR3L									
H'FEC8	DR4H									
H'FEC9	DR4L									
H'FECA	DR5H									
H'FECB	DR5L									
H'FECC	DR6H									
H'FECD	DR6L									
H'FECE	DR7H									
H'FECF	DR7L									
H'FED0	DR8H									
H'FED1	DR8L									
H'FED2	DR9H									
H'FED3	DR9L									
H'FED4	DR10H									

Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FED5	DR10L									ISP
H'FED6	DR11H									
H'FED7	DR11L									
H'FED8	DR12H									
H'FED9	DR12L									
H'FEDA	DR13H									
H'FEDB	DR13L									
H'FEDC	DR14H									
H'FEDD	DR14L									
H'FEDE	DR15H									
H'FEDF	DR15L									
H'FEE0	DR16H									
H'FEE1	DR16L									
H'FEE2	DR17H									
H'FEE3	DR17L									
H'FEE4	DR18H									
H'FEE5	DR18L									
H'FEE6	DR19H									
H'FEE7	DR19L									
H'FEE8	DR20H									
H'FEE9	DR20L									
H'FEEA	DR21H									
H'FEEB	DR21L									
H'FEEC	DR22H									
H'FEED	DR22L									
H'FEEE	DR23H									
H'FEEF	DR23L									
H'FEF0	DR24H									
H'FEF1	DR24L									
H'FEF2	DR25H									

Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEF3	DR25L									ISP
H'FEF4	DR26H									
H'FEF5	DR26L									
H'FEF6	DR27H									
H'FEF7	DR27L									
H'FEF8	DR28H									
H'FEF9	DR28L									
H'FEFA	DR29H									
H'FEFB	DR29L									
H'FEFC	DR30H									
H'FEFD	DR30L									
H'FEFE	DR31H									
H'FEFF	DR31L									
H'FF00	—	—	—	—	—	—	—	—	—	—
H'FF01	—	—	—	—	—	—	—	—	—	
H'FF02	—	—	—	—	—	—	—	—	—	
H'FF03	—	—	—	—	—	—	—	—	—	
H'FF04	—	—	—	—	—	—	—	—	—	
H'FF05	—	—	—	—	—	—	—	—	—	
H'FF06	—	—	—	—	—	—	—	—	—	
H'FF07	—	—	—	—	—	—	—	—	—	
H'FF08	—	—	—	—	—	—	—	—	—	
H'FF09	—	—	—	—	—	—	—	—	—	
H'FF0A	—	—	—	—	—	—	—	—	—	
H'FF0B	—	—	—	—	—	—	—	—	—	
H'FF0C	—	—	—	—	—	—	—	—	—	
H'FF0D	—	—	—	—	—	—	—	—	—	
H'FF0E	—	—	—	—	—	—	—	—	—	
H'FF0F	—	—	—	—	—	—	—	—	—	

Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF10	—	—	—	—	—	—	—	—	—	—
H'FF11	—	—	—	—	—	—	—	—	—	
H'FF12	—	—	—	—	—	—	—	—	—	
H'FF13	—	—	—	—	—	—	—	—	—	
H'FF14	—	—	—	—	—	—	—	—	—	
H'FF15	—	—	—	—	—	—	—	—	—	
H'FF16	—	—	—	—	—	—	—	—	—	
H'FF17	—	—	—	—	—	—	—	—	—	
H'FF18	IPR									ISP
H'FF19	ICSR	—	—	IRST	FRSTT	FNS3	FNS2	FNS1	FNS0	
H'FF1A	—	—	—	—	—	—	—	—	—	—
H'FF1B	—	—	—	—	—	—	—	—	—	
H'FF1C	—	—	—	—	—	—	—	—	—	
H'FF1D	—	—	—	—	—	—	—	—	—	
H'FF1E	—	—	—	—	—	—	—	—	—	
H'FF1F	—	—	—	—	—	—	—	—	—	
H'FF20	—	—	—	—	—	—	—	—	—	
H'FF21	—	—	—	—	—	—	—	—	—	
H'FF22	—	—	—	—	—	—	—	—	—	
H'FF23	SYSCR8	$IOF0_7E$	$IOF0_6E$	$IOF0_5E$	$IOF0_4E$	$IOF0_3E$	$IOF0_2E$	$IOF0_1E$	$IOF0_0E$	Port 8
H'FF24	SYSCR9	$IOF1_7E$	$IOF1_6E$	$IOF1_5E$	$IOF1_4E$	$IOF1_3E$	$IOF1_2E$	$IOF1_1E$	$IOF1_0E$	Port 9
H'FF25	SYSCR10	$IOF2_7E$	$IOF2_6E$	$IOF2_5E$	$IOF2_4E$	$IOF2_3E$	$IOF2_2E$	$IOF2_1E$	$IOF2_0E$	Port 10
H'FF26	—	—	—	—	—	—	—	—	—	—
H'FF27	—	—	—	—	—	—	—	—	—	
H'FF28	FEDGE	$FEDGE_7$	$FEDGE_6$	$FEDGE_5$	$FEDGE_4$	$FEDGE_3$	$FEDGE_2$	$FEDGE_1$	$FEDGE_0$	ISP
H'FF29	REDGE	$REDGE_7$	$REDGE_6$	$REDGE_5$	$REDGE_4$	$REDGE_3$	$REDGE_2$	$REDGE_1$	$REDGE_0$	
H'FF2A	—	—	—	—	—	—	—	—	—	—
H'FF2B	—	—	—	—	—	—	—	—	—	
H'FF2C	P1DDR	$P1_7DDR$	—	—	—	$P1_3DDR$	$P1_2DDR$	$P1_1DDR$	$P1_0DDR$	Port 1

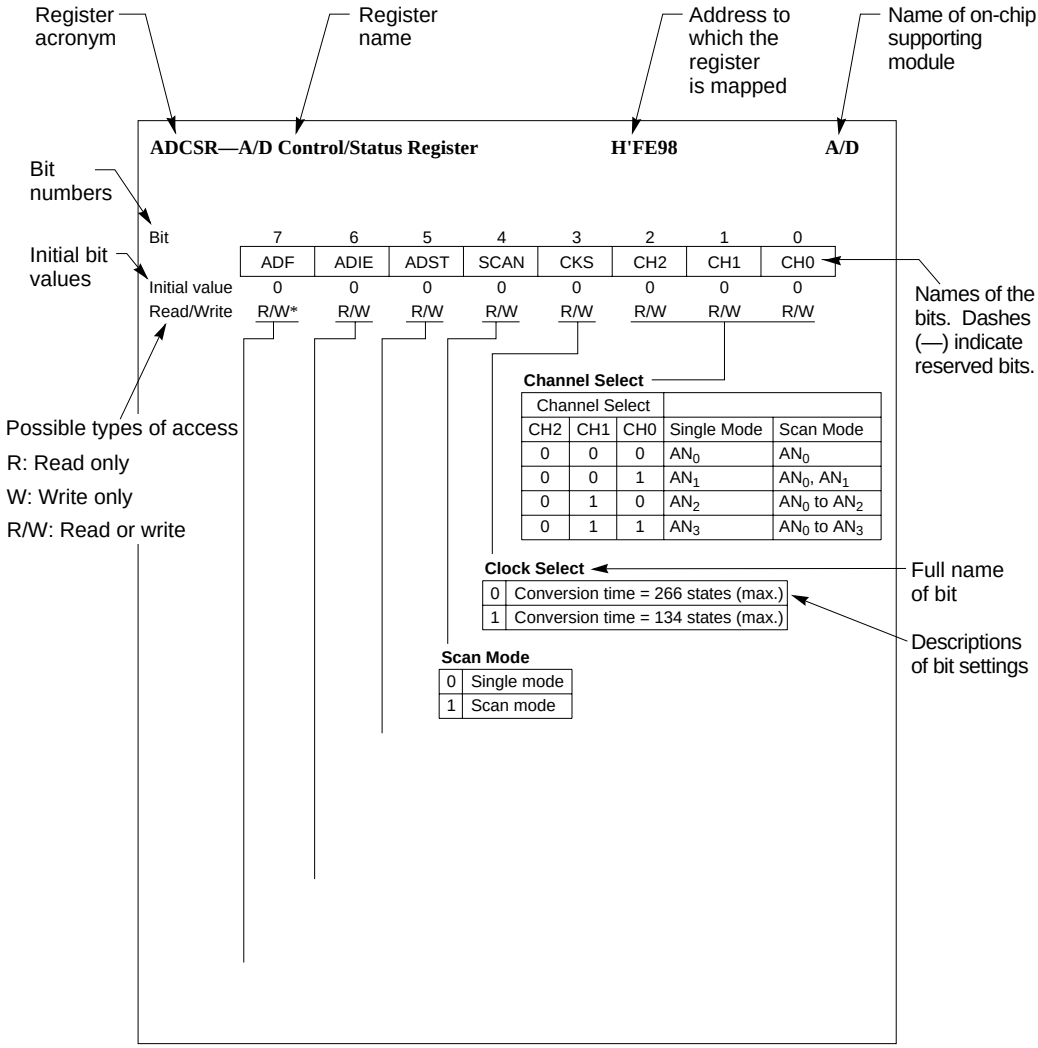
Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF2D	—	—	—	—	—	—	—	—	—	—
H'FF2E	—	—	—	—	—	—	—	—	—	
H'FF2F	—	—	—	—	—	—	—	—	—	
H'FF30	P5DDR	$P5_7$DDR	$P5_6$DDR	—	$P5_4$DDR	$P5_3$DDR	$P5_2$DDR	$P5_1$DDR	$P5_0$DDR	Port 5
H'FF31	P6DDR	—	—	$P6_5$DDR	$P6_4$DDR	$P6_3$DDR	$P6_2$DDR	$P6_1$DDR	$P6_0$DDR	Port 6
H'FF32	—	—	—	—	—	—	—	—	—	—
H'FF33	P8DDR	$P8_7$DDR	$P8_6$DDR	$P8_5$DDR	$P8_4$DDR	$P8_3$DDR	$P8_2$DDR	$P8_1$DDR	$P8_0$DDR	Port 8
H'FF34	P9DDR	$P9_7$DDR	$P9_6$DDR	$P9_5$DDR	$P9_4$DDR	$P9_3$DDR	$P9_2$DDR	$P9_1$DDR	$P9_0$DDR	Port 9
H'FF35	P10DDR	$P10_7$DDR	$P10_6$DDR	$P10_5$DDR	$P10_4$DDR	$P10_3$DDR	$P10_2$DDR	$P10_1$DDR	$P10_0$DDR	Port 10
H'FF36	P11DDR	$P11_7$DDR	$P11_6$DDR	$P11_5$DDR	$P11_4$DDR	$P11_3$DDR	$P11_2$DDR	$P11_1$DDR	$P11_0$DDR	Port 11
H'FF37	P12DDR	$P12_7$DDR	$P12_6$DDR	$P12_5$DDR	$P12_4$DDR	$P12_3$DDR	$P12_2$DDR	$P12_1$DDR	$P12_0$DDR	Port 12
H'FF38	—	—	—	—	—	—	—	—	—	—
H'FF39	—	—	—	—	—	—	—	—	—	
H'FF3A	—	—	—	—	—	—	—	—	—	
H'FF3B	—	—	—	—	—	—	—	—	—	
H'FF3C	—	—	—	—	—	—	—	—	—	
H'FF3D	—	—	—	—	—	—	—	—	—	
H'FF3E	—	—	—	—	—	—	—	—	—	
H'FF3F	—	—	—	—	—	—	—	—	—	
H'FF40	IPRA	—	IRQ_0			—	PWM			INTC
H'FF41	IPRB	—	ISP			—	ISP			
H'FF42	IPRC	—	ISP			—	ISP			
H'FF43	IPRD	—	SCI			—	A/D			
H'FF44	DTEA	—	—	—	IRQ_0	—	OCF_2	OCF_1	OCF_0	
H'FF45	DTEB	—	ISFL6	ISFL5	ISFL4	—	ISFL2	ISFL1	ISFL0	
H'FF46	DTEC	—	ISFH6	ISFH5	ISFH4	—	ISFH2	ISFH1	ISFH0	
H'FF47	DTED	—	TXI	RXI	—	—	—	—	ADI	
H'FF48	WCR	—	—	BCC1	BCC0	WMS1	WMS0	WC1	WC0	WSC

Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF49	RAMCR	RAME	IBE	—	—	—	—	—	—	RAM
H'FF4A	MDCR	—	—	—	—	—	MDS2	MDS1	MDS0	For mode select
H'FF4B	SBYCR	SSBY	—	—	—	—	—	—	—	
H'FF4C	SYSCR1	—	—	IRQ_0E	NMIEG	BRLE	—	—	—	Port 1
H'FF4D	—	—	—	—	—	—	—	—	—	—
H'FF4E	Write controller									WDT
H'FF4F	RSTCSR	WRST	RSTOE	—	—	—	—	—	—	
H'FF50	—	—	—	—	—	—	—	—	—	—
H'FF51	—	—	—	—	—	—	—	—	—	
H'FF52	—	—	—	—	—	—	—	—	—	
H'FF53	—	—	—	—	—	—	—	—	—	
H'FF54	—	—	—	—	—	—	—	—	—	
H'FF55	—	—	—	—	—	—	—	—	—	
H'FF56	—	—	—	—	—	—	—	—	—	
H'FF57	—	—	—	—	—	—	—	—	—	
H'FF58	—	—	—	—	—	—	—	—	—	
H'FF59	—	—	—	—	—	—	—	—	—	
H'FF5A	—	—	—	—	—	—	—	—	—	
H'FF5B	—	—	—	—	—	—	—	—	—	
H'FF5C	—	—	—	—	—	—	—	—	—	
H'FF5D	—	—	—	—	—	—	—	—	—	
H'FF5E	—	—	—	—	—	—	—	—	—	
H'FF5F	—	—	—	—	—	—	—	—	—	
H'FF60	—	—	—	—	—	—	—	—	—	
H'FF61	—	—	—	—	—	—	—	—	—	
H'FF62	—	—	—	—	—	—	—	—	—	
H'FF63	—	—	—	—	—	—	—	—	—	
H'FF64	—	—	—	—	—	—	—	—	—	
H'FF65	—	—	—	—	—	—	—	—	—	

Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF66	—	—	—	—	—	—	—	—	—	—
H'FF67	—	—	—	—	—	—	—	—	—	
H'FF68	—	—	—	—	—	—	—	—	—	
H'FF69	—	—	—	—	—	—	—	—	—	
H'FF6A	—	—	—	—	—	—	—	—	—	
H'FF6B	—	—	—	—	—	—	—	—	—	
H'FF6C	—	—	—	—	—	—	—	—	—	
H'FF6D	—	—	—	—	—	—	—	—	—	
H'FF6E	—	—	—	—	—	—	—	—	—	
H'FF6F	—	—	—	—	—	—	—	—	—	
H'FF70	—	—	—	—	—	—	—	—	—	
H'FF71	—	—	—	—	—	—	—	—	—	
H'FF72	—	—	—	—	—	—	—	—	—	
H'FF73	—	—	—	—	—	—	—	—	—	
H'FF74	—	—	—	—	—	—	—	—	—	
H'FF75	—	—	—	—	—	—	—	—	—	
H'FF76	—	—	—	—	—	—	—	—	—	
H'FF77	—	—	—	—	—	—	—	—	—	
H'FF78	—	—	—	—	—	—	—	—	—	
H'FF79	—	—	—	—	—	—	—	—	—	
H'FF7A	—	—	—	—	—	—	—	—	—	
H'FF7B	—	—	—	—	—	—	—	—	—	
H'FF7C	—	—	—	—	—	—	—	—	—	
H'FF7D	—	—	—	—	—	—	—	—	—	
H'FF7E	—	—	—	—	—	—	—	—	—	
H'FF7F	—	—	—	—	—	—	—	—	—	

Note: For abbreviations, refer to appendix G, Abbreviations Listing.

B.2 Register Descriptions


Register acronym
Register name
Address to which the register is mapped
Name of on-chip supporting module
ADCSR—A/D Control/Status Register
H'FE98
A/D
Bit numbers
Initial bit values
Bit
Initial value
Read/Write
7 6 5 4 3 2 1 0
ADF ADIE ADST SCAN CKS CH2 CH1 CH0
0 0 0 0 0 0 0 0
R/W* R/W R/W R/W R/W R/W R/W R/W
Names of the bits. Dashes (—) indicate reserved bits.
Possible types of access
R: Read only
W: Write only
R/W: Read or write
Channel Select
Channel Select
CH2 CH1 CH0 Single Mode Scan Mode
0 0 0 AN0 AN0
0 0 1 AN1 AN0, AN1
0 1 0 AN2 AN0 to AN2
0 1 1 AN3 AN0 to AN3
Clock Select
0 Conversion time = 266 states (max.)
1 Conversion time = 134 states (max.)
Full name of bit
Descriptions of bit settings
Scan Mode
0 Single mode
1 Scan mode

ADDRn (H)—A/D Data Register n (High) **n = A (H'FE80), B (H'FE82), C (H'FE84), D (H'FE86)** **A/D**

Bit	7	6	5	4	3	2	1	0
	AD_9	AD_8	AD_7	AD_6	AD_5	AD_4	AD_3	AD_2
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R

Upper 8 bits of 10-bit A/D conversion result

ADDRn (L)—A/D Data Register n (Low) **n = A (H'FE81), B (H'FE83), C (H'FE85), D (H'FE87)** **A/D**

Bit	7	6	5	4	3	2	1	0
	AD_1	AD_0	—	—	—	—	—	—
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R

Lower 2 bits of 10-bit A/D conversion result

Bit	7	6	5	4	3	2	1	0
	ADF	ADIE	ADST	SCAN	CKS	CH2	CH1	CH0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W*	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Channel Select

Channel Select				
CH2	CH1	CH0	Single Mode	Scan Mode
0	0	0	AN_0	AN_0
0	0	1	AN_1	AN_0, AN_1
0	1	0	AN_2	AN_0 to AN_2
0	1	1	AN_3	AN_0 to AN_3

Clock Select

0	Conversion time = 266 states (max.)
1	Conversion time = 134 states (max.)

Scan Mode

0	Single mode
1	Scan mode

A/D Start

0	A/D conversion is halted.
1	• During single mode: One A/D conversion is performed, then bit is automatically cleared to 0. • During scan mode: A/D conversion starts and continues cyclically on all selected channels until 0 is written to this bit.

A/D Interrupt Enable

0	A/D-end interrupt is disabled.
1	A/D-end interrupt is enabled.

A/D End Flag

0	Cleared to 0 when: • The MCU is reset or enters a standby mode • CPU reads ADF = 1, then writes 0 into this bit • ADI interrupt is served by DTC
1	Set to 1 when in: • Single mode: At the completion of A/D conversion. • Scan mode: After all selected channels have been converted.

Note: * For write, only writing 0 is enabled.

Bit	7	6	5	4	3	2	1	0
	TRGE	—	—	—	—	—	—	—
Initial value	0	1	1	1	1	1	1	1
Read/Write	R/W	—	—	—	—	—	—	—

Trigger Enable

0	The A/D external trigger is disabled.
1	The A/D external trigger is enabled and $P4_0$ is set for input. A/D conversion starts on the falling edge of $\overline{\text{ADTRG}}$ input at $P4_0$.

Bit	7	6	5	4	3	2	1	0
	OVF	WT/$\overline{IT}$	TME	—	—	CKS2	CKS1	CKS0
Initial value	0	0	0	1	1	0	0	0
Read/Write	R/(W)*1	R/W	R/W	—	—	R/W	R/W	R/W

Clock Select

0	0	0	ø/2 (51.2 µs)*2
0	0	1	ø/32 (819.2 µs)
0	1	0	ø/64 (1.6 ms)
0	1	1	ø/128 (3.3 ms)
1	0	0	ø/256 (6.6 ms)
1	0	1	ø/512 (13.1 ms)
1	1	0	ø/2048 (52.4 ms)
1	1	1	ø/4096 (104.9 ms)

Timer Enable

0	Timer is disabled. TCNT is initialized to H'00 and stopped.
1	Timer is enabled. TCNT starts incrementing and the CPU interrupt request is enabled.

Timer Mode Select

0	Interval timer mode (IRQ_0 interrupt request)
1	Watchdog timer mode (Reset)

Overflow Flag

0	Cleared to 0 if the CPU reads OVF as 1.
1	Set to 1 when TCNT changes from H'FF to H'00.

Notes: 1. For write, only writing 0 is enabled.
2. The time in parentheses is the time it takes for TCNT to increment from H'00 to H'FF and change to H'00 again when ø = 10 MHz.

TCNT—Timer Counter **H'FE8B** **WDT**

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Count value

P1DR—Port 1 Data Register **H'FE8C** **Port 1**

Bit	7	6	5	4	3	2	1	0
	$P1_7$	—	—	—	$P1_3$	$P1_2$	$P1_1$	$P1_0$
Initial value	0	—	—	—	0	0	0	—
Read/Write	R/W	—	—	—	R/W	R/W	R/W	R

P5DR—Port 5 Data Register **H'FE90** **Port 5**

Bit	7	6	5	4	3	2	1	0
	$P5_7$	$P5_6$	—	$P5_4$	$P5_3$	$P5_2$	$P5_1$	$P5_0$
Initial value	0	—	—	0	0	0	0	0
Read/Write	R/W	R	—	R/W	R/W	R/W	R/W	R/W

P6DR—Port 6 Data Register **H'FE91** **Port 6**

Bit	7	6	5	4	3	2	1	0
	—	—	$P6_5$	$P6_4$	$P6_3$	$P6_2$	$P6_1$	$P6_0$
Initial value	1	1	0	0	0	0	0	0
Read/Write	—	—	R/W	R/W	R/W	R/W	R/W	R/W

P7DR—Port 7 Data Register **H'FE92** **Port 7**

Bit	7	6	5	4	3	2	1	0
	$P7_7$	$P7_6$	$P7_5$	$P7_4$	$P7_3$	$P7_2$	$P7_1$	$P7_0$
Initial value	—	—	—	—	—	—	—	—
Read/Write	R	R	R	R	R	R	R	R

P8DR—Port 8 Data Register **H'FE93** **Port 8**

Bit	7	6	5	4	3	2	1	0
	$P8_7$	$P8_6$	$P8_5$	$P8_4$	$P8_3$	$P8_2$	$P8_1$	$P8_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P9DR—Port 9 Data Register **H'FE94** **Port 9**

Bit	7	6	5	4	3	2	1	0
	$P9_7$	$P9_6$	$P9_5$	$P9_4$	$P9_3$	$P9_2$	$P9_1$	$P9_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P10DR—Port 10 Data Register **H'FE95** **Port 10**

Bit	7	6	5	4	3	2	1	0
	$P10_7$	$P10_6$	$P10_5$	$P10_4$	$P10_3$	$P10_2$	$P10_1$	$P10_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P11DR—Port 11 Data Register **H'FE96** **Port 11**

Bit	7	6	5	4	3	2	1	0
	$P11_7$	$P11_6$	$P11_5$	$P11_4$	$P11_3$	$P11_2$	$P11_1$	$P11_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P12DR—Port 12 Data Register **H'FE97** **Port 12**

Bit	7	6	5	4	3	2	1	0
	$P12_7$	$P12_6$	$P12_5$	$P12_4$	$P12_3$	$P12_2$	$P12_1$	$P12_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
	C/$\overline{A}$	CHR	PE	O/$\overline{E}$	STOP	—	CKS1	CKS0
Initial value	0	0	0	0	0	1	0	0
Read/Write	R/(W)	R/W	R/W	R/W	R/W	—	R/W	R/W

Clock Select

0	0	ø clock
0	1	ø/4 clock
1	0	ø/16 clock
1	1	ø/64 clock

Stop Bit Length

0	One stop bit
1	Two stop bits

Parity Mode

0	Even parity
1	Odd parity

Parity Enable

0	Transmit: No parity bit added. Receive: Parity bit not checked.
1	Transmit: Parity bit added. Receive: Parity bit checked.

Character Length

0	8-bit data length
1	7-bit data length

Communication Mode

0	Asynchronous
1	Synchronous

BRR—Bit Rate Register **H'FE99** **SCI**

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Constant that determines the baud rate

Bit	7	6	5	4	3	2	1	0
	TIE	RIE	TE	RE	—	—	CKE1	CKE0
Initial value	0	0	0	0	1	1	0	0
Read/Write	R/W	R/W	R/W	R/W	—	—	R/W	R/W

Clock Enable 0

0	SCK pin is not used.
1	SCK pin is used for output.

Clock Enable 1

0	Internal clock
1	External clock, input at SCK pin

Receive Enable

0	Receive disabled
1	Receive enabled

Transmit Enable

0	Transmit disabled
1	Transmit enabled

Receive Interrupt Enable

0	Receive interrupt request (RXI) is disabled.
1	Receive interrupt request (RXI) is enabled.

Transmit Interrupt Enable

0	Transmit interrupt request (TXI) is disabled.
1	Transmit interrupt request (TXI) is enabled.

TDR—Transmit Data Register **H'FE9B** **SCI**

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Transmit data

Bit	7	6	5	4	3	2	1	0
	TDRE	RDRF	ORER	FER	PER	—	—	—
Initial value	1	0	0	0	0	1	1	1
Read/Write	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	—	—	—

Parity Error

0	Cleared to 0 when: • CPU reads PER as 1 • The MCU is reset or enters a standby mode
1	Set to 1 when a parity error occurs (parity of receive data does not match parity selected by O/$\overline{E}$ bit).

Framing Error

0	Cleared to 0 when: • CPU reads FER as 1 • The MCU is reset or enters a stand by mode
1	Set to 1 when a framing error occurs (stop bit is 0).

Overrun Error

0	Cleared to 0 when: • CPU reads ORER as 1 • The MCU is reset or enters a standby mode
1	Set to 1 when an overrun error occurs (next data is completely received while RDRF bit is set at 1).

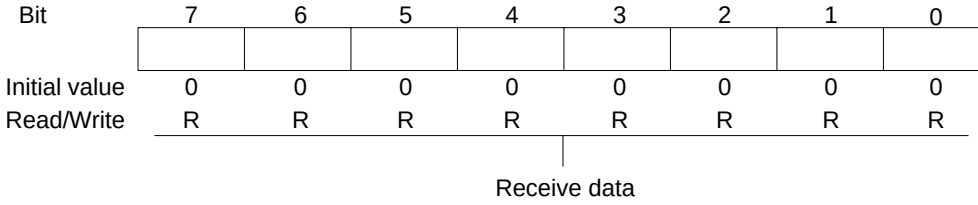
Receive Data Register Full

0	Cleared from 1 to 0 when: • CPU reads RDRF as 1 • RDR is read by the DTC • The MCU is reset or enters a standby mode
1	Set to 1 when one character is received normally and transferred from RSR to RDR.

Transmit Data Register Empty

0	Cleared to 0 when: • CPU reads TDRE as 1 • The DTC writes data into TDR
1	Set to 1 when: • The MCU is reset or enters a standby mode • Data is transferred from TDR to TSR • TE is cleared to 0 when this bit is 0

Note: * For write, only writing 0 is enabled.


Bit
7
6
5
4
3
2
1
0
Initial value
0
0
0
0
0
0
0
0
Read/Write
R
R
R
R
R
R
R
R
Receive data

Bit	7	6	5	4	3	2	1	0
	TCE	$\overline{\text{FRM}}$	CKS1	CKS0	OMS	OE2	OE1	OE0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Output Mode Select
Output Enable 2 to 0

Bit 3	TSR/TXR Source						Transfer Condition
OMS	ODL5	ODL4	ODL3	ODL2	ODL1	ODL0	
0	—	—	—	—	—	OD00	TMR = OCR0
	—	—	—	—	OD01	—	TMR = OCR1
	—	—	—	OD02	—	—	TMR = OCR2
1	OD05	OD04	OD03	OD02	OD01	OD00	TMR = OCR0
	OD15	OD14	OD13	OD12	OD11	OD10	TMR = OCR1
	OD25	OD24	OD23	OD22	OD21	OD20	TMR = OCR2

Bit 3	Bit 2	Bit 1	Bit 0						
OMS	OE2	OE1	OE0	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0
0	0	0	0	—	—	—	—	—	—
	0	0	1	—	—	—	—	—	Output
	0	1	0	—	—	—	—	Output	Output
	0	1	1	—	—	—	Output	Output	Output
	1	*	*	—	—	—	Output	Output	Output
1	0	0	0	—	—	—	—	—	—
	0	0	1	—	—	—	—	—	Output
	0	1	0	—	—	—	—	Output	Output
	0	1	1	—	—	—	Output	Output	Output
	1	0	0	—	—	Output	Output	Output	Output
	1	0	1	—	Output	Output	Output	Output	Output
	1	1	*	Output	Output	Output	Output	Output	Output

Clock Select

0	0	ø clock
0	1	ø/4 clock
1	0	ø/8 clock
1	1	ø/16 clock

Free-Running Mode

0	TMR functions as a free-running timer counting from H'0000 to H'FFFF.
1	At a match between TMR and OCR0, TMR is cleared to H'0000 at the next clock timing.

Timer Count Enable

0	TMR stops counting.
1	TMR counts up.

Bit	7	6	5	4	3	2	1	0
	TRE2	TRE0	OCIE2	OCIE1	OCIE0	OCF2	OCF1	OCF0
Initial value	0	0	0	0	0	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/(W)	R/(W)	R/(W)

Output Compare Flag 0

0	This bit is cleared to 0 when: • The CPU reads the OCF0 bit, then clears it to 0 • The data transfer controller (DTC) is activated by setting OCF0
1	This bit is set to 1 when TMR = OCR0 and the next count clock is input to TMR.

Output Compare Flag 1

0	This bit is to 0 when: • The CPU reads the OCF1 bit, then clears it to 0 • The data transfer controller (DTC) is activated by setting OCF1
1	This bit is set to 1 when TMR = OCR1 and the next count clock is input to TMR.

Output Compare Flag 2

0	This bit is to 0 when: • The CPU reads the OCF2 bit, then clears it to 0 • The data controller (DTC) is activated by setting OCF2
1	This bit is set to 1 when TMR = OCR2 and the next count clock is input to TMR.

Output Compare Interrupt Enable 0

0	Output compare interrupt request 0 (OCI0) is disabled.
1	Output compare interrupt request 0 (OCI0) is enabled.

Output Compare Interrupt Enable 1

0	Output compare interrupt request 1 (OCI1) is disabled.
1	Output compare interrupt request 1 (OCI1) is enabled.

Output Compare Interrupt Enable 2

0	Output compare interrupt request 2 (OCI2) is disabled.
1	Output compare interrupt request 2 (OCI2) is enabled.

Transfer Request Enable 0

0	The PWM cannot set interconnection flag high 6 (ICFH6) in the ISP.
1	ICFH6 is set to 1 when output compare flag 0 (OCF0) is set to 1.

Transfer Request Enable 1

0	The PWM cannot set interconnection flag high 7 (ICFH7) in the ISP.
1	ICFH7 is set to 1 when output compare flag 2 (OCF2) is set to 1.

ODL—Output Data Latch **H'FEA2** **PWM**

Bit	7	6	5	4	3	2	1	0
	—	—	ODL5	ODL4	ODL3	ODL2	ODL1	ODL0
Initial value	1	1	0	0	0	0	0	0
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W

Stores the PWM output value

ODR0—Output Data Register 0 **H'FEA3** **PWM**

Bit	7	6	5	4	3	2	1	0
	—	—	OD05	OD04	OD03	OD02	OD01	OD00
Initial value	1	1	0	0	0	0	0	0
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W

Stores the output code for transferring to the output data latch (ODL)

ODR1—Output Data Register 1 **H'FEA4** **PWM**

Bit	7	6	5	4	3	2	1	0
	—	—	OD15	OD14	OD13	OD12	OD11	OD10
Initial value	1	1	0	0	0	0	0	0
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W

ODR2—Output Data Register 2 **H'FEA5** **PWM**

Bit	7	6	5	4	3	2	1	0
	—	—	OD25	OD24	OD23	OD22	OD21	OD20
Initial value	1	1	0	0	0	0	0	0
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W

OCR0 (H and L)—Output Compare Register 0 **H'FEA6 and H'FEA7 (High and Low)** **PWM**

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Continually compared with the value in TMR.
When a match is detected, a value in the output data register (ODR) corresponding to each OCR number is transferred to an output data latch (ODL).

OCR1 (H and L)—Output Compare Register 1 **H'FEA8 and H'FEA9 (High and Low)** **PWM**

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

OCR2 (H and L)—Output Compare Register 2 **H'FEAA and H'FEAB (High and Low)** **PWM**

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TMR (H and L)—Timer (High and Low) **H'FEAC and H'FEAD** **PWM**

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Count value

ISFH—Interrupt Status Flags High **H'FEB0** **ISP**

Bit	7	6	5	4	3	2	1	0
	ISFH7	ISFH6	ISFH5	ISFH4	ISFH3	ISFH2	ISFH1	ISFH0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)

Interrupt status flags

ISFL—Interrupt Status Flags Low **H'FEB1** **ISP**

Bit	7	6	5	4	3	2	1	0
	ISFL7	ISFL6	ISFL5	ISFL4	ISFL3	ISFL2	ISFL1	ISFL0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)

Interrupt status flags

IOF2—I/O Flags 2 **H'FEB2** **ISP**

Bit	7	6	5	4	3	2	1	0
	IOF27	IOF26	IOF25	IOF24	IOF23	IOF22	IOF21	IOF20
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R

IOF1—I/O Flags 1 **H'FEB3** **ISP**

Bit	7	6	5	4	3	2	1	0
	IOF17	IOF16	IOF15	IOF14	IOF13	IOF12	IOF11	IOF10
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R

IOF0—I/O Flags 0 **H'FEB4** **ISP**

Bit	7	6	5	4	3	2	1	0
	IOF07	IOF06	IOF05	IOF04	IOF03	IOF02	IOF01	IOF00
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R

EGF—Edge Detection Flags **H'FEB5** **ISP**

Bit	7	6	5	4	3	2	1	0
	EGF7	EGF6	EGF5	EGF4	EGF3	EGF2	EGF1	EGF0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R

ICFH—Interconnection Flags High **H'FEB6** **ISP**

Bit	7	6	5	4	3	2	1	0
	ICFH7	ICFH6	ICFH5	ICFH4	ICFH3	ICFH2	ICFH1	ICFH0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

ICFL—Interconnection Flags Low **H'FEB7** **ISP**

Bit	7	6	5	4	3	2	1	0
	ICFL7	ICFL6	ICFL5	ICFL4	ICFL3	ICFL2	ICFL1	ICFL0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

IEFH—Interrupt Enable Flags High **H'FEB8** **ISP**

Bit	7	6	5	4	3	2	1	0
	IEFH7	IEFH6	IEFH5	IEFH4	IEFH3	IEFH2	IEFH1	IEFH0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Interrupt Enable

0	Interrupt requests from ISFH to the CPU are disabled.
1	Interrupt requests from ISFH to the CPU are enabled.

IEFL—Interrupt Enable Flags Low **H'FEB9** **ISP**

Bit	7	6	5	4	3	2	1	0
	IEFL7	IEFL6	IEFL5	IEFL4	IEFL3	IEFL2	IEFL1	IEFL 0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Interrupt Enable

0	Interrupt requests from ISFL to the CPU are disabled.
1	Interrupt requests from ISFL to the CPU are enabled.

IOIEH—I/O Interrupt Enable Flags High **H'FEBA** **ISP**

Bit	7	6	5	4	3	2	1	0
	IOIEH7	IOIEH6	IOIEH5	IOIEH4	IOIEH3	IOIEH2	IOIEH1	IOIEH0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

I/O Interrupt Enable

0	ISFH is not set by IOF2.
1	ISFH is set if IOF2 is read and then cleared to 0.

IOIEL—I/O Interrupt Enable Flags Low **H'FEBB** **ISP**

Bit	7	6	5	4	3	2	1	0
	IOIEL7	IOIEL6	IOIEL5	IOIEL4	IOIEL3	IOIEL2	IOIEL1	IOIEL0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

I/O Interrupt Enable

0	ISFL is not set by IOF1.
1	ISFL is set if IOF1 is set.

CLEH—Read Clear Enable Register High **H'FEBC** **ISP**

Bit	7	6	5	4	3	2	1	0
	CLEH7	CLEH6	CLEH5	CLEH4	CLEH3	CLEH2	CLEH1	CLEH0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Read Clear Enable

0	ICFH is not cleared when it is read.
1	ICFH is automatically cleared when it is read.

CLEL—Read Clear Enable Register Low **H'FEBD** **ISP**

Bit	7	6	5	4	3	2	1	0
	CLEL7	CLEL6	CLEL5	CLEL4	CLEL3	CLEL2	CLEL1	CLEL0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Read Clear Enable

0	ICFL is not cleared when it is read.
1	ICFL is automatically cleared when it is read.

EVER—Event Input Enable Register **H'FEBF** **ISP**

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	EVE2	EVE1	EVE0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R/W	R/W	R/W

Event Input Enable

0	Event input is disabled.
1	Event input is enabled.

DRn (H and L)—Data Register n (High and Low) **(n = 0 to 31)**
H'FEC0 to H'FEFE **ISP**

Bit	7	6	5	4	3	2	1	0
Initial value	—	—	—	—	—	—	—	—
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

IPR—ISP Page Register **H'FF18** **ISP**

Bit	7	6	5	4	3	2	1	0
Initial value	—	—	—	—	—	—	—	—
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Page address of ISP memory access instructions

Bit	7	6	5	4	3	2	1	0
	—	—	IRST	FRSTT	FNS3	FNS2	FNS1	FNS0
Initial value	0	0	1	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Operation Selection

Bit 5 IRST	Bit 4 FRSTT	Bit 3 FNS3	Bit 2 FNS2	Bit 1 FNS1	Bit 0 FNS0	Operation
1	*	*	*	*	*	ISP reset
0	0	*	*	*	*	Normal operation
0	1	0 to 1	0 to 0	0 to 1	0 to 1	Function restart
0	1	1	1	*	*	Normal operation

Bit	7	6	5	4	3	2	1	0
	$IOF0_7E$	$IOF0_6E$	$IOF0_5E$	$IOF0_4E$	$IOF0_3E$	$IOF0_2E$	$IOF0_1E$	$IOF0_0E$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Pin Function Selection

0	Port 8
1	IOF0

Bit	7	6	5	4	3	2	1	0
	$IOF1_7E$	$IOF1_6E$	$IOF1_5E$	$IOF1_4E$	$IOF1_3E$	$IOF1_2E$	$IOF1_1E$	$IOF1_0E$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Pin Function Selection

0	Port 9
1	IOF1

Bit	7	6	5	4	3	2	1	0
	$IOF2_7E$	$IOF2_6E$	$IOF2_5E$	$IOF2_4E$	$IOF2_3E$	$IOF2_2E$	$IOF2_1E$	$IOF2_0E$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Pin Function Selection

0	Port 10
1	IOF2

FEDGE and REDGE—Falling Edge Enable Register and Rising Edge Enable Register H'FF28 and H'FF29 ISP

FEDGE

Bit	7	6	5	4	3	2	1	0
	$FEDGE_7$	$FEDGE_6$	$FEDGE_5$	$FEDGE_4$	$FEDGE_3$	$FEDGE_2$	$FEDGE_1$	$FEDGE_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

REDGE

Bit	7	6	5	4	3	2	1	0
	$REDGE_7$	$REDGE_6$	$REDGE_5$	$REDGE_4$	$REDGE_3$	$REDGE_2$	$REDGE_1$	$REDGE_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

$IOF2_n$ Pin Function Selection

$REDGE_n$	$FEDGE_n$	
0	0	Level input pin
1	0	Rising-edge input pin
0	1	Falling-edge input pin
1	1	Rising-and-falling-edge input pin

P1DDR—Port 1 Data Direction Register H'FF2C Port 1

Bit	7	6	5	4	3	2	1	0
	$P1_7DDR$	—	—	—	$P1_3DDR$	$P1_2DDR$	$P1_1DDR$	$P1_0DDR$
Initial value	0	1	1	1	0	0	0	1
Read/Write	W	—	—	—	W	W	W	W

Port 1 Input/Output Selection

0	Input port
1	Output port

P5DDR—Port 5 Data Direction Register H'FF30

Bit	7	6	5	4	3	2	1	0
	$P5_7DDR$	$P5_6DDR$	—	$P5_4DDR$	$P5_3DDR$	$P5_2DDR$	$P5_1DDR$	$P5_0DDR$
Initial value	0	0	1	0	0	0	0	0
Read/Write	W	W	—	W	W	W	W	W

Port 5 Input/Output Selection

0	Input port
1	Output port

P6DDR—Port 6 Data Direction Register H'FF31

Bit	7	6	5	4	3	2	1	0
	—	—	$P6_5DDR$	$P6_4DDR$	$P6_3DDR$	$P6_2DDR$	$P6_1DDR$	$P6_0DDR$
Initial value	1	1	0	0	0	0	0	0
Read/Write	—	—	W	W	W	W	W	W

Port 6 Input/Output Selection

0	Input port
1	Output port

P8DDR—Port 8 Data Direction Register H'FF33

Bit	7	6	5	4	3	2	1	0
	$P8_7DDR$	$P8_6DDR$	$P8_5DDR$	$P8_4DDR$	$P8_3DDR$	$P8_2DDR$	$P8_1DDR$	$P8_0DDR$
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 8 Input/Output Selection

0	Input port
1	Output port

P9DDR—Port 9 Data Direction Register H'FF34 Port 9

Bit	7	6	5	4	3	2	1	0
	$P9_7DDR$	$P9_6DDR$	$P9_5DDR$	$P9_4DDR$	$P9_3DDR$	$P9_2DDR$	$P9_1DDR$	$P9_0DDR$
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 9 Input/Output Selection

0	Input port
1	Output port

P10DDR—Port 10 Data Direction Register H'FF35 Port 10

Bit	7	6	5	4	3	2	1	0
	$P10_7DDR$	$P10_6DDR$	$P10_5DDR$	$P10_4DDR$	$P10_3DDR$	$P10_2DDR$	$P10_1DDR$	$P10_0DDR$
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 10 Input/Output Selection

0	Input port
1	Output port

P11DDR—Port 11 Data Direction Register H'FF36 Port 11

Bit	7	6	5	4	3	2	1	0
	$P11_7DDR$	$P11_6DDR$	$P11_5DDR$	$P11_4DDR$	$P11_3DDR$	$P11_2DDR$	$P11_1DDR$	$P11_0DDR$
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 11 Input/Output Selection

0	Input port
1	Output port

Bit	7	6	5	4	3	2	1	0
	$P12_7DDR$	$P12_6DDR$	$P12_5DDR$	$P12_4DDR$	$P12_3DDR$	$P12_2DDR$	$P12_1DDR$	$P12_0DDR$
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 12 Input/Output Selection

0	Input port
1	Output port

IPRA—Interrupt Priority Register A H'FF40 INTC

Bit	7	6	5	4	3	2	1	0
	—				—			
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R/W	R/W	R/W	R	R/W	R/W	R/W
		[IRQ_0]				[PWM]		
		Specifies priority levels 7 to 0				Specifies priority levels 7 to 0		

IPRB—Interrupt Priority Register B H'FF41 INTC

Bit	7	6	5	4	3	2	1	0
	—				—			
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R/W	R/W	R/W	R	R/W	R/W	R/W
		[ISP (ISFL7 to ISFL4)]				[ISP (ISFL3 to ISFL0)]		
		Specifies priority levels 7 to 0				Specifies priority levels 7 to 0		

IPRC—Interrupt Priority Register C H'FF42 INTC

Bit	7	6	5	4	3	2	1	0
	—				—			
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R/W	R/W	R/W	R	R/W	R/W	R/W
		[ISP (ISFH7 to ISFH4)]				[ISP (ISFH3 to ISFH0)]		
		Specifies priority levels 7 to 0				Specifies priority levels 7 to 0		

IPRD—Interrupt Priority Register D **H'FF43** **INTC**

Bit	7	6	5	4	3	2	1	0
	—				—			
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R/W	R/W	R/W	R	R/W	R/W	R/W
		[SCI]				[A/D converter]		
		Specifies priority levels 7 to 0				Specifies priority levels 7 to 0		

DTEA—DT Enable Register A **H'FF44** **INTC**

Bit	7	6	5	4	3	2	1	0
	—	—	—		—			
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

OCF0

0	Normal interrupt
1	DTC start

OCF1

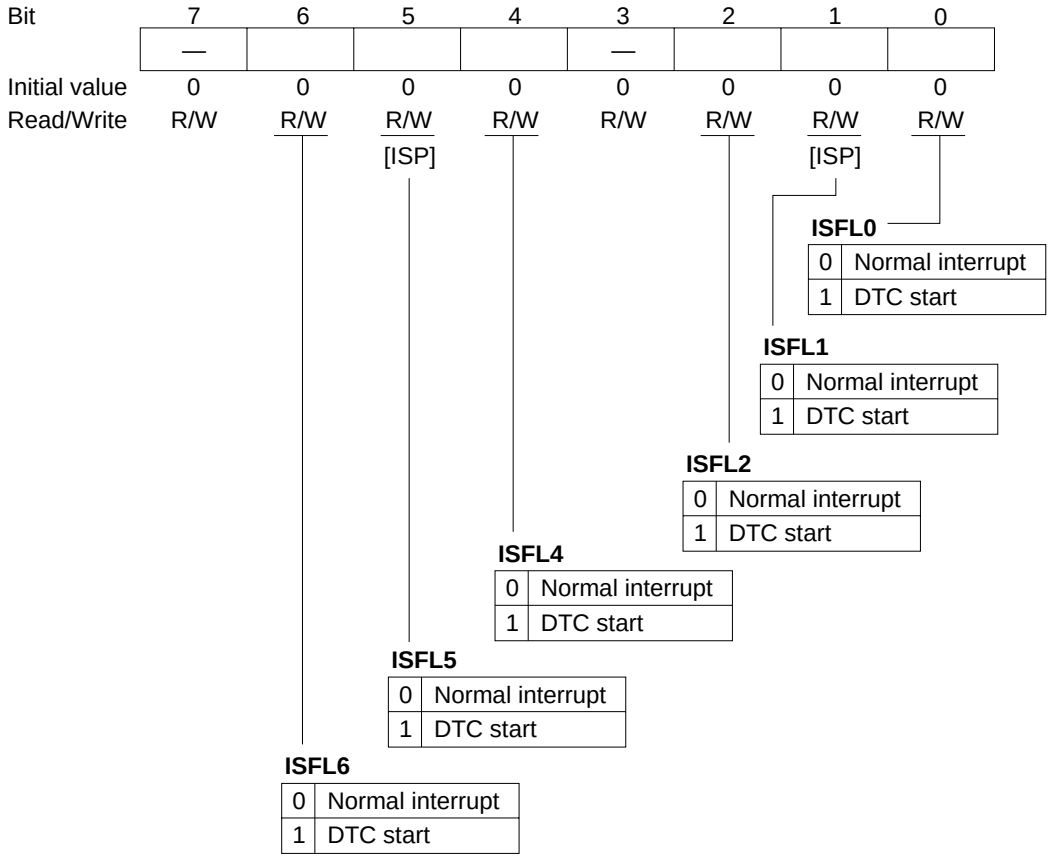
0	Normal interrupt
1	DTC start

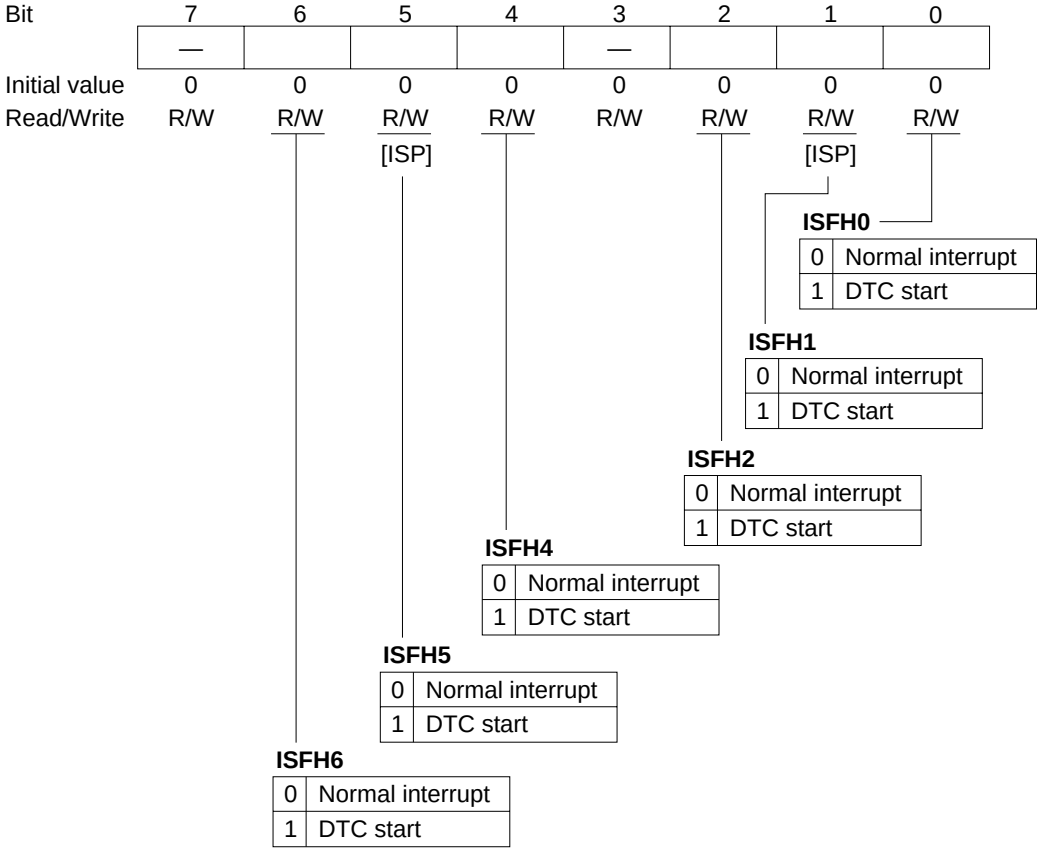
OCF2

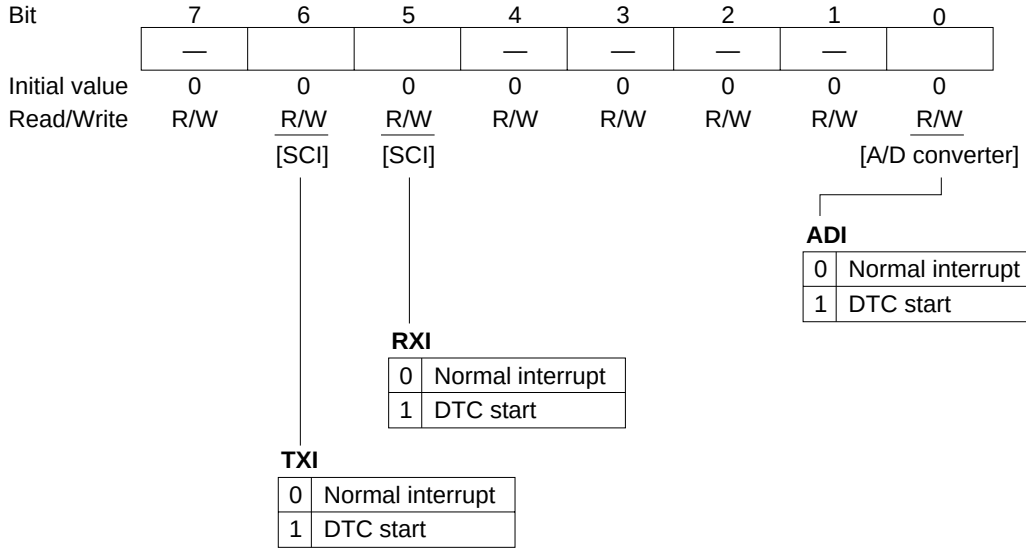
0	Normal interrupt
1	DTC start

IRQ_0

0	Normal interrupt
1	DTC start


Bit
7
6
5
4
3
2
1
0
—
—
Initial value
0 0 0 0 0 0 0 0
Read/Write
R/W R/W R/W R/W R/W R/W R/W R/W
[ISP]
[ISP]
ISFL0
0 Normal interrupt
1 DTC start
ISFL1
0 Normal interrupt
1 DTC start
ISFL2
0 Normal interrupt
1 DTC start
ISFL4
0 Normal interrupt
1 DTC start
ISFL5
0 Normal interrupt
1 DTC start
ISFL6
0 Normal interrupt
1 DTC start


Bit
7
6
5
4
3
2
1
0
—
—
Initial value
0
0
0
0
0
0
0
0
Read/Write
R/W
R/W
R/W
R/W
R/W
R/W
R/W
R/W
[ISP]
[ISP]
ISFH0
0 Normal interrupt
1 DTC start
ISFH1
0 Normal interrupt
1 DTC start
ISFH2
0 Normal interrupt
1 DTC start
ISFH4
0 Normal interrupt
1 DTC start
ISFH5
0 Normal interrupt
1 DTC start
ISFH6
0 Normal interrupt
1 DTC start


Bit
7
6
5
4
3
2
1
0
—
—
—
—
Initial value
0
0
0
0
0
0
0
0
Read/Write
R/W
R/W
R/W
R/W
R/W
R/W
R/W
R/W
[SCI]
[SCI]
[A/D converter]
ADI
0 Normal interrupt
1 DTC start
RXI
0 Normal interrupt
1 DTC start
TXI
0 Normal interrupt
1 DTC start

WCR—Wait-State Control Register **H'FF48** **WSC**

Bit	7	6	5	4	3	2	1	0
	—	—	BCC1	BCC0	WMS1	WMS0	WC1	WC0
Initial value	1	1	0	0	0	0	1	1
Read/Write	—	—	R/W	R/W	R/W	R/W	R/W	R/W

Wait Count 1 and 0

0	0	No wait states (T_W) are inserted.
0	1	1 wait state is inserted.
1	0	2 wait states are inserted.
1	1	3 wait states are inserted.

Wait Mode Select 1 and 0

0	0	Programmable wait mode
0	1	No wait states are inserted, regardless of the wait count.
1	0	Pin wait mode
1	1	Pin auto-wait mode

Bus Cycle Control 1 and 0

BCC1	BCC0	Page 0	Page 1	Pages 2 and 3	Pages 4 to 15
0	0	3 states			
0	1	2 states	3 states		
1	0	2 states		3 states	
1	1	2 states			3 states

RAMCR—RAM Control Register **H'FF49** **RAM**

Bit	7	6	5	4	3	2	1	0
	RAME	IBE	—	—	—	—	—	—
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	—	—	—	—	—	—

ISP Bus Enable

0	Bus access right of the ISP is disabled.
1	Bus access right of the ISP is enabled.

RAM Enable

0	RAM is disabled.
1	RAM is enabled.

MDCR—Mode Control Register **H'FF4A** **For Mode Selection**

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	MDS2	MDS1	MDS0
Initial value	1	1	0	0	0	—*	—*	—*
Read/Write	—	—	—	—	—	R	R	R

Mode Select
Values of the mode pins (MD_2 to MD_0)

Note: * Initialized according to MD_2 to MD_0.

SBYCR—Software Standby Control Register **H'FF4B** **For Mode Selection**

Bit	7	6	5	4	3	2	1	0
	SSBY	—	—	—	—	—	—	—
Initial value	0	1	1	1	1	1	1	1
Read/Write	R/W	—	—	—	—	—	—	—

Software Standby

0	SLEEP instruction causes a transition to the sleep mode.
1	SLEEP instruction causes a transition to the software standby mode.

Bit	7	6	5	4	3	2	1	0
	—	—	IRQ_0E	NMIEG	BRLE	—	—	—
Initial value	1	0	0	0	0	1	1	1
Read/Write	R	R/W	R/W	R/W	R/W	R	R	—

Bus Release Enable

0	$P1_2$ and $P1_1$ function as input/output pins.
1	$P1_2$ functions as the $\overline{BREQ}$ input pin. $P1_1$ functions as the $\overline{BACK}$ output pin.

Nonmaskable Interrupt Edge

0	A nonmaskable interrupt is generated on the falling edge of the input at the NMI pin.
1	A nonmaskable interrupt is generated on the rising edge of the input at the NMI pin.

Interrupt Request 0 Enable

0	$P5_4$ functions as an input/output pin.
1	$P5_4$ functions as the $\overline{IRQ_0}$ input pin.

RSTCSR—Reset Control/Status Register **H'FF4F** **WDT**

Bit	7	6	5	4	3	2	1	0
	WRST	RSTOE	—	—	—	—	—	—
Initial value	0	0	1	1	1	1	1	1
Read/Write	R/(W)*	R/W	—	—	—	—	—	—

Reset Output Enable

0	The reset signal generated by TCNT overflow is not output to the external devices.
1	The reset signal generated by TCNT overflow is output to the external devices.

Watchdog Timer Reset

0	This bit is cleared to 0 when a reset signal is input from the $\overline{RES}$ pin, or when the software writes a 0 in this bit.
1	This bit is set to 1 when TCNT overflows in the watchdog timer mode and a reset signal is generated.

Note: * By software, only a 0 can be written to clear the flag.

Appendix C I/O Port Schematic Diagrams

C.1 Schematic Diagram of Port 1

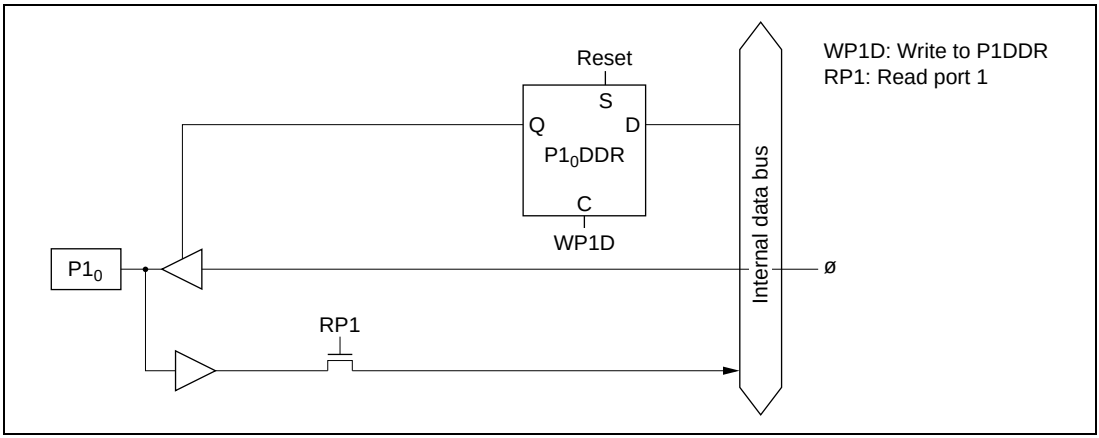


Figure C.1 (a) Schematic Diagram of Port 1, Pin P1$_0$

Table C.1 (a) Port 1 Port Read (Pin P1$_0$)

Setting	Port Read Data
DDR = 0	Pin value
DDR = 1	ø

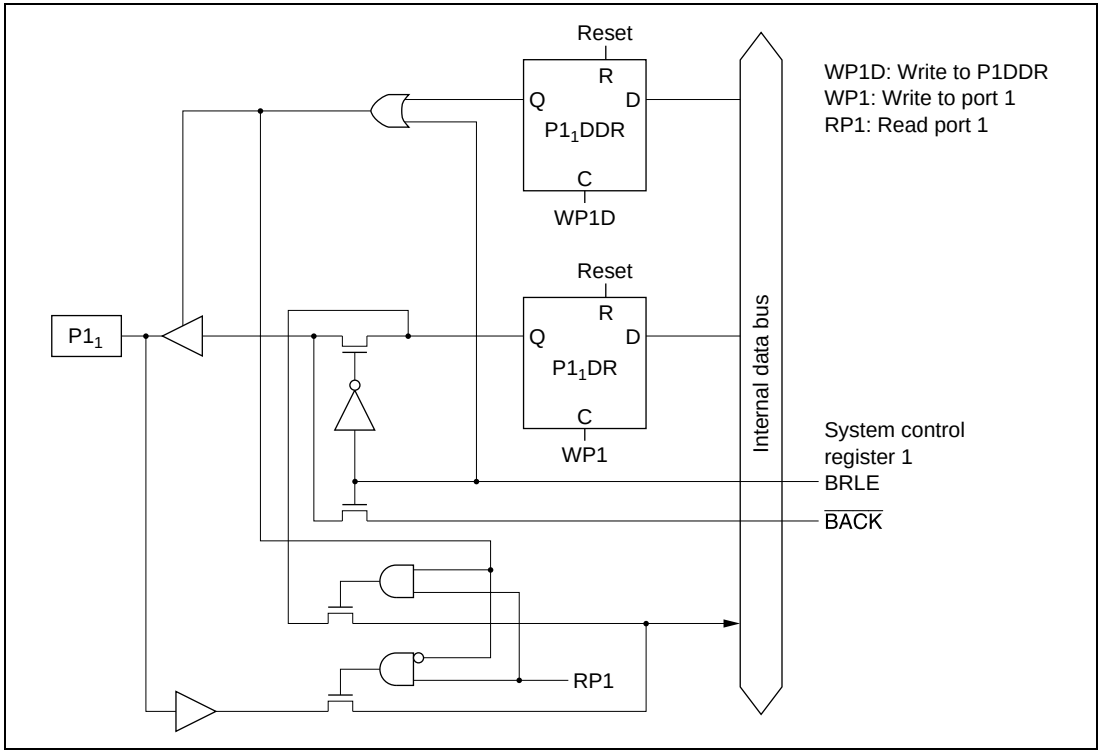


Figure C.1 (b) Schematic Diagram of Port 1, Pin $P1_1$

Table C.1 (b) Port 1 Port Read (Pin $P1_1$)

Setting		Port Read Data
BRLE = 1		DR value
BRLE = 0	DDR = 0	Pin value
	DDR = 1	DR value

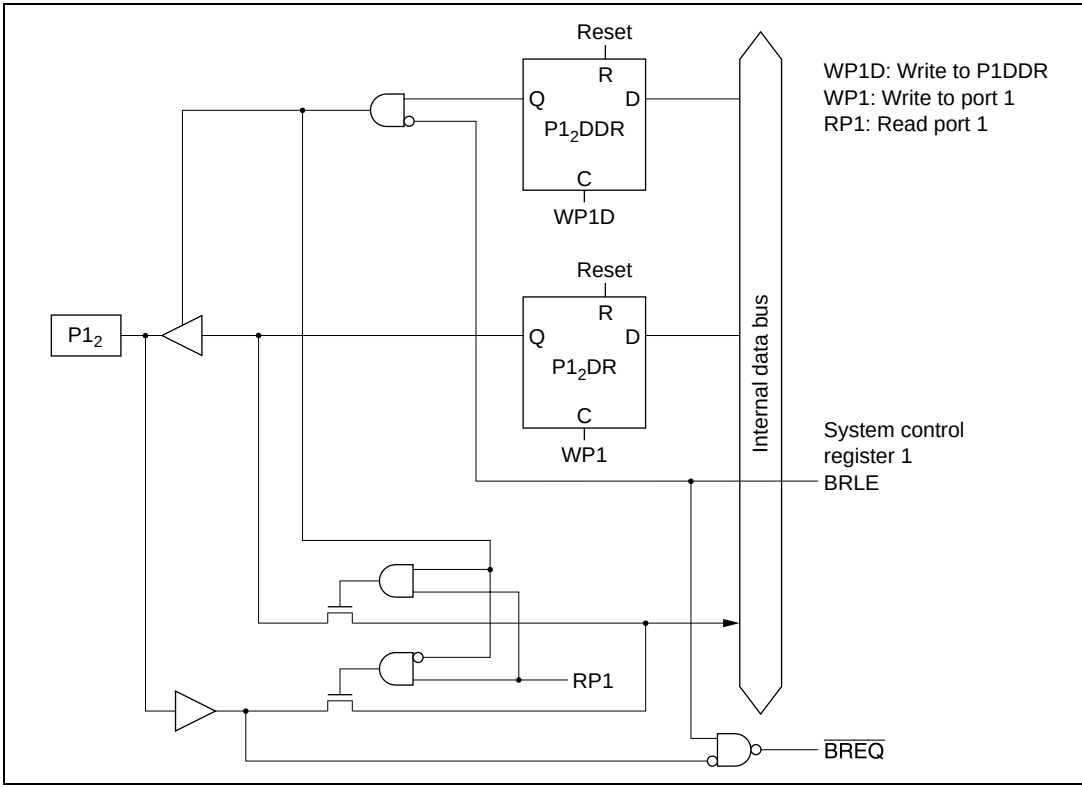


Figure C.1 (c) Schematic Diagram of Port 1, Pin P1$_2$

Table C.1 (c) Port 1 Port Read (Pin P1$_2$)

Setting		Port Read Data
BRLE = 1		Pin value
BRLE = 0	DDR = 0	Pin value
	DDR = 1	DR value

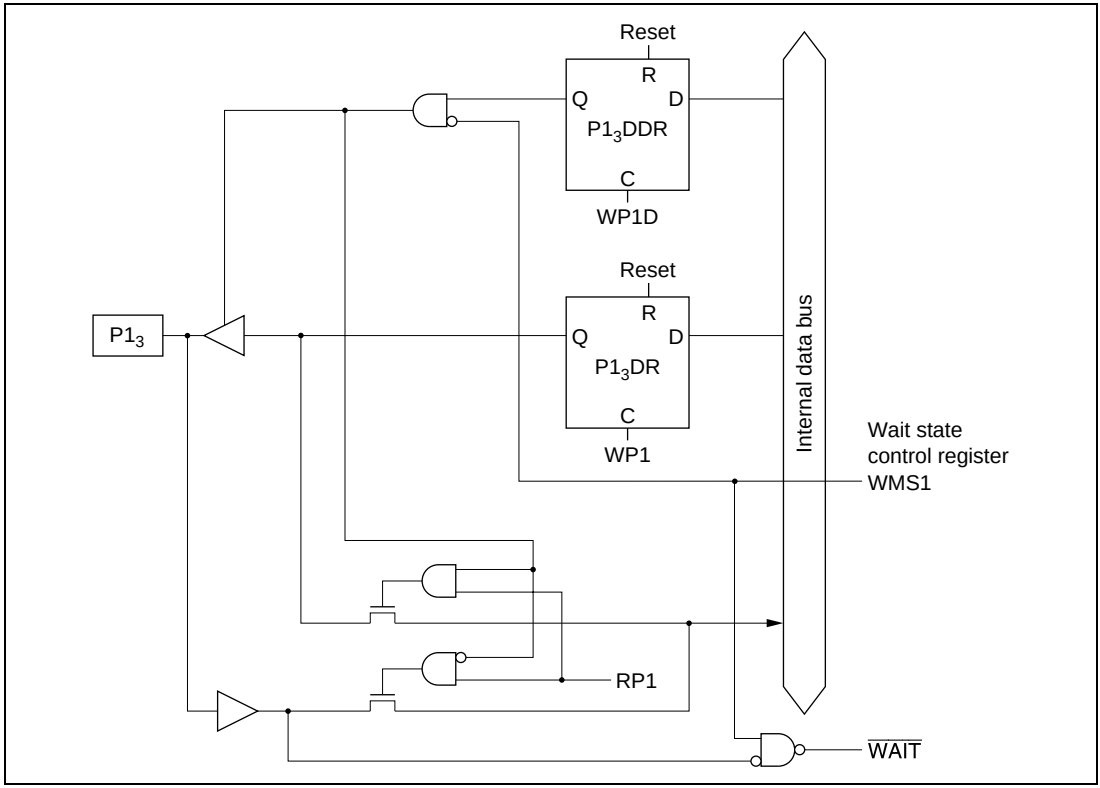


Figure C.1 (d) Schematic Diagram of Port 1, Pin P1$_3$

Table C.1 (d) Port 1 Port Read (Pin P1$_3$)

Setting		Port Read Data
WMS1 = 1		DR value
WMS = 0	DDR = 0	Pin value
	DDR = 1	DR value

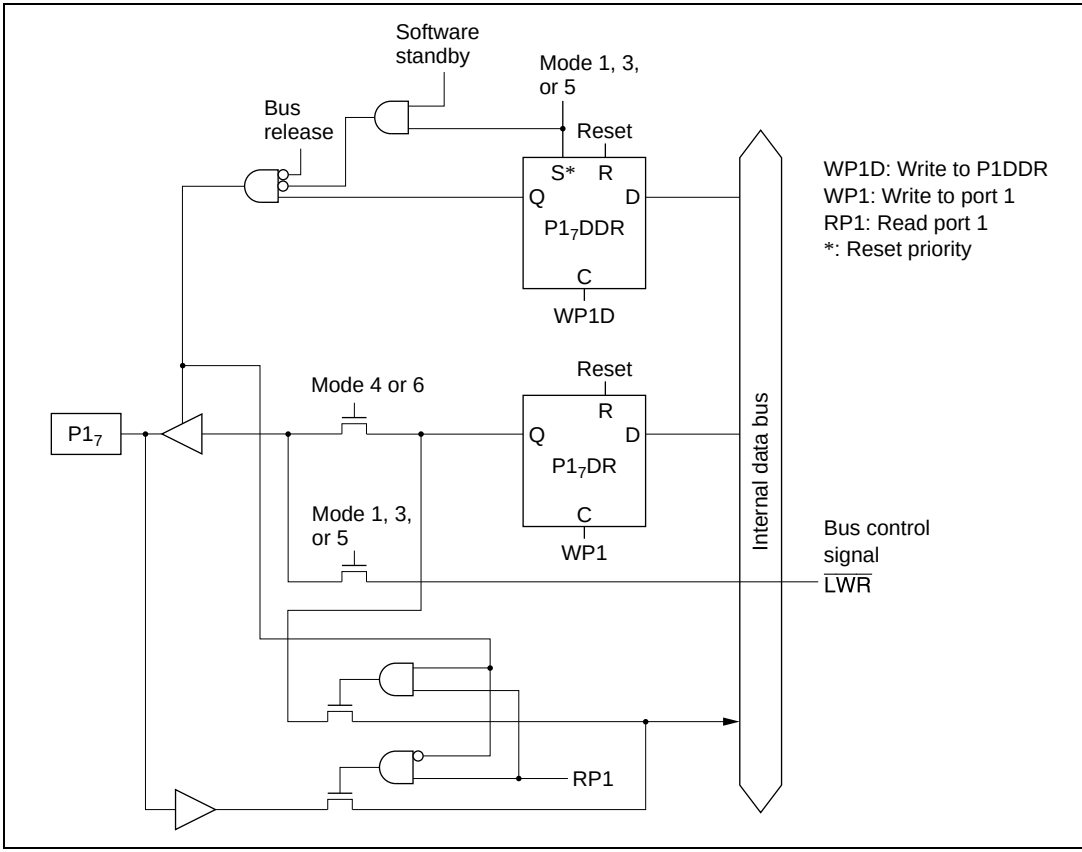


Figure C.1 (e) Schematic Diagram of Port 1, Pin P1$_7$

Table C.1 (e) Port 1 Port Read (Pin P1$_7$)

Mode		Port Read Data
1, 3, 5		DR value
4, 6	DDR = 0	Pin value
	DDR = 1	DR value

C.2 Schematic Diagram of Port 5

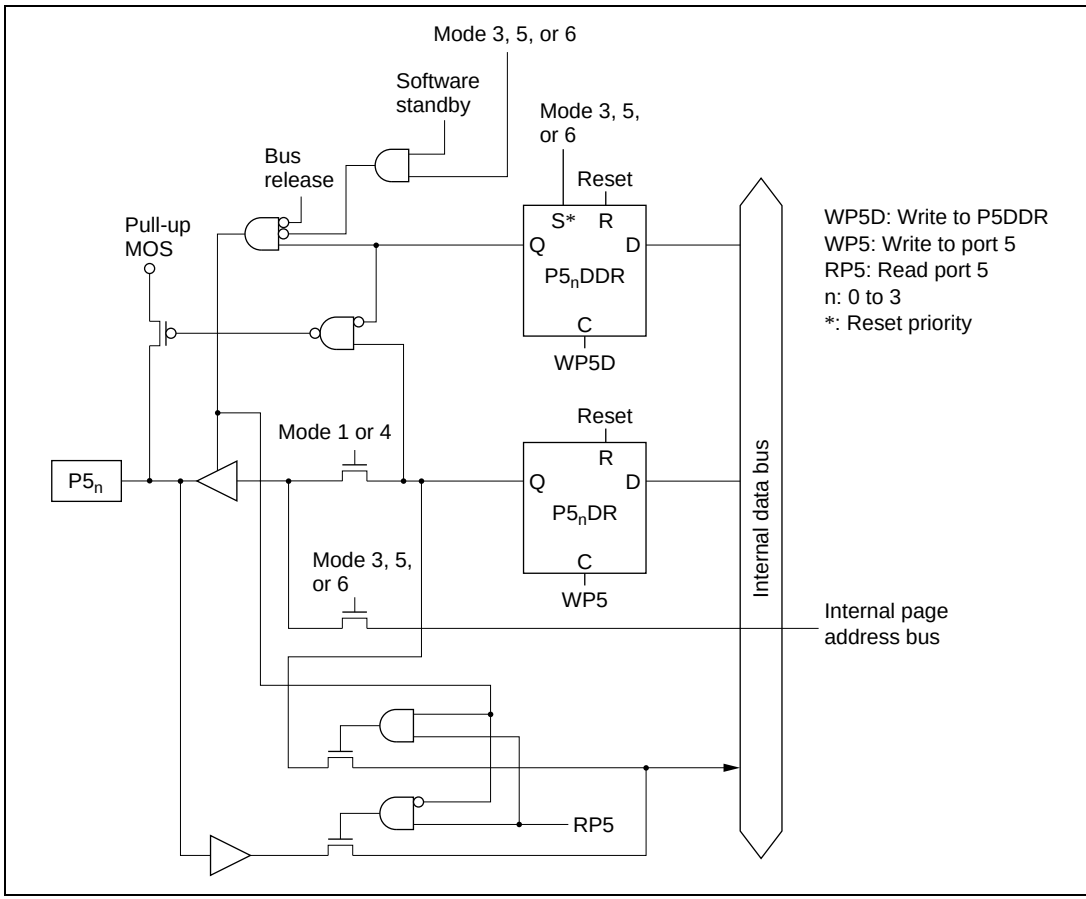


Figure C.2 (a) Schematic Diagram of Port 5, Pins $P5_0$, $P5_1$, $P5_2$, and $P5_3$

Table C.2 (a) Port 5 Port Read (Pins $P5_0$, $P5_1$, $P5_2$, and $P5_3$)

Mode		Port Read Data
1, 4	DDR = 0	Pin value
	DDR = 1	DR value
3, 5, 6		DR value

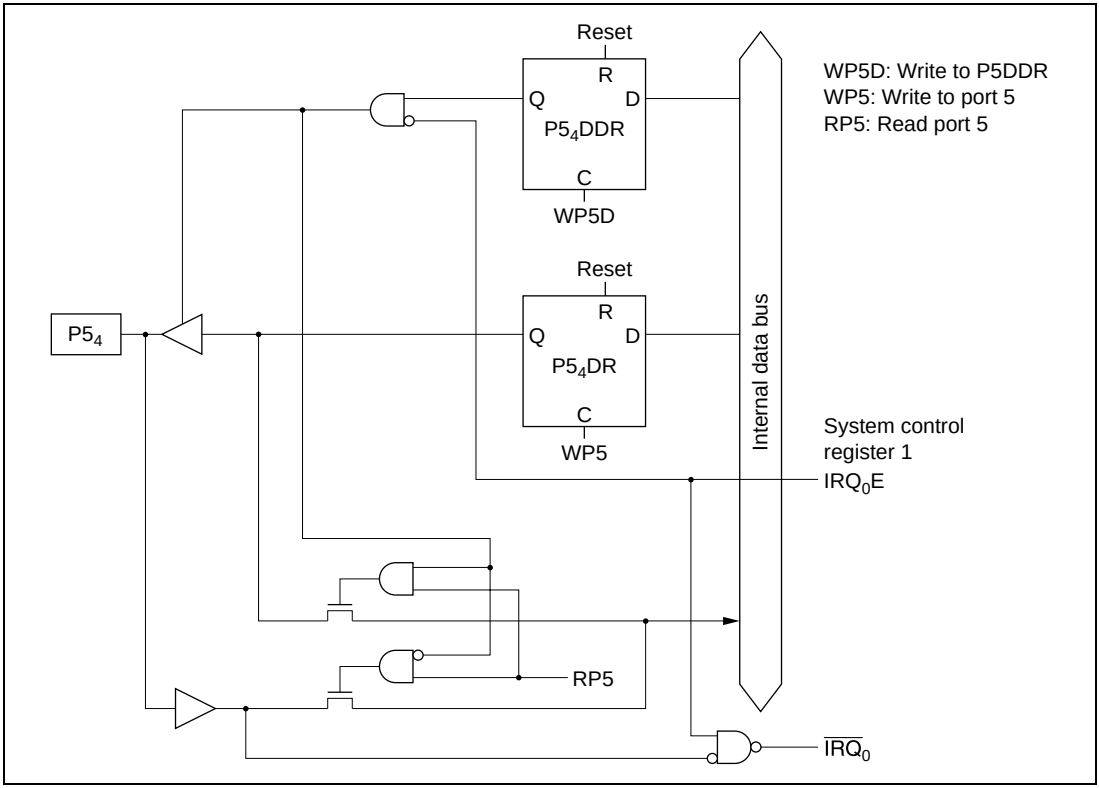


Figure C.2 (b) Schematic Diagram of Port 5, Pin P5$_4$

Table C.2 (b) Port 5 Port Read (Pin P5$_4$)

Setting		Port Read Data
IRQ$_0$E = 1		Pin value
IRQ$_0$E = 0	DDR = 0	Pin value
	DDR = 1	DR value

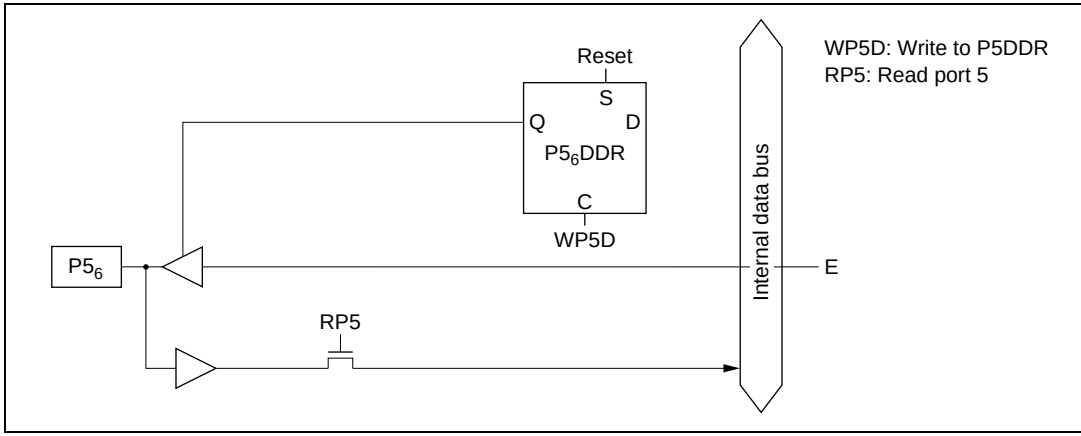


Figure C.2 (c) Schematic Diagram of Port 5, Pin P5$_6$

Table C.2 (c) Port 5 Port Read (Pin P5$_6$)

Setting	Port Read Data
DDR = 0	Pin value
DDR = 1	E

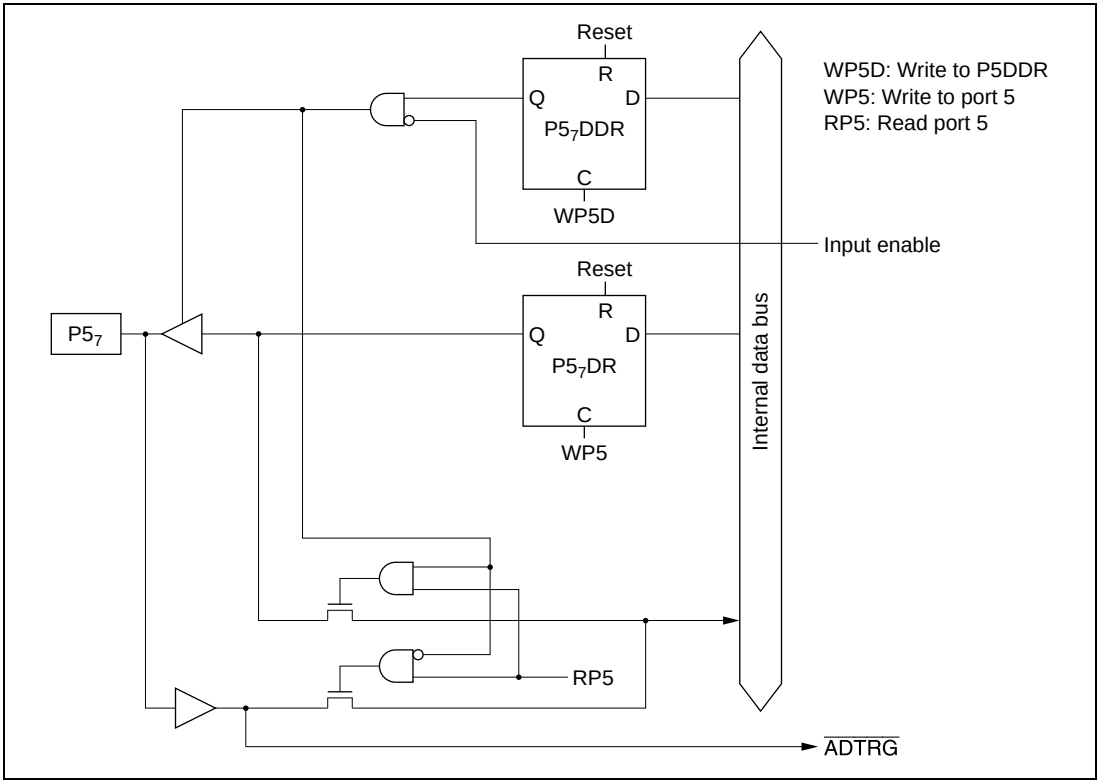


Figure C.2 (d) Schematic Diagram of Port 5, Pin $P5_7$

Table C.2 (d) Port 5 Port Read (Pin $P5_7$)

Setting		Port Read Data
Input enable		A/D converter trigger input value
Input disable	DDR = 0	Pin value
	DDR = 1	DR value

C.3 Schematic Diagram of Port 6

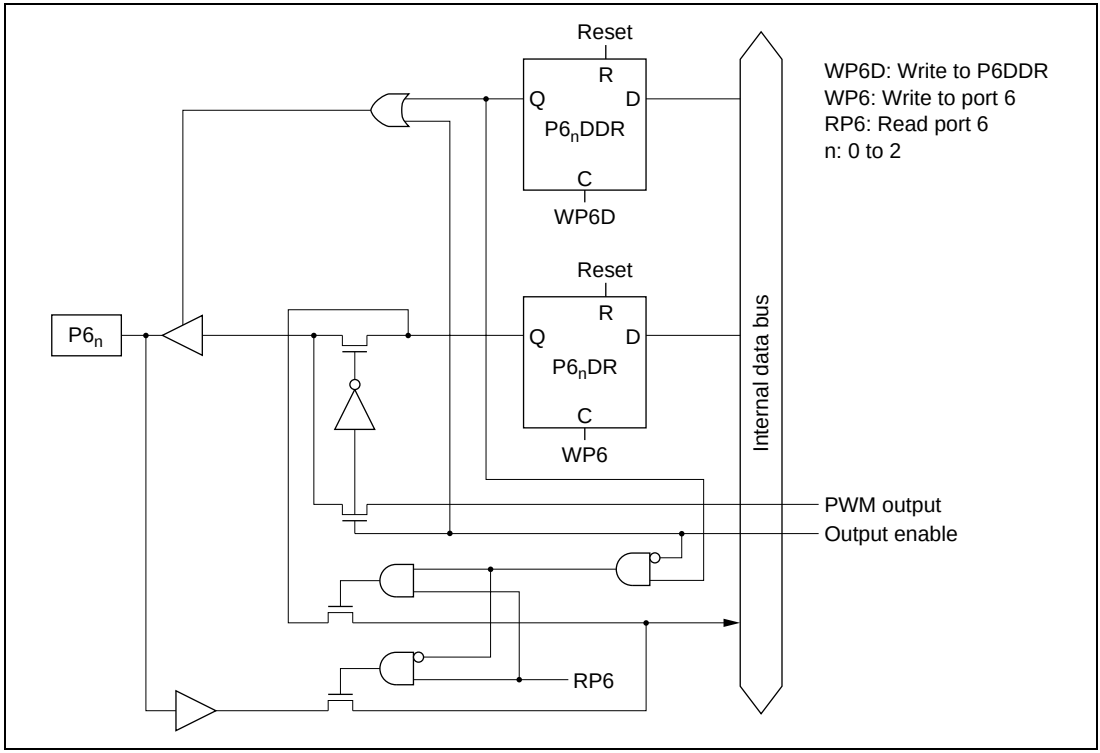


Figure C.3 (a) Schematic Diagram of Port 6, Pins $P6_0$, $P6_1$, and $P6_2$

Table C.3 (a) Port 6 Port Read (Pins $P6_0$, $P6_1$, and $P6_2$)

Setting		Port Read Data
Output enable		Pin value
Output disable	DDR = 0	Pin value
	DDR = 1	DR value

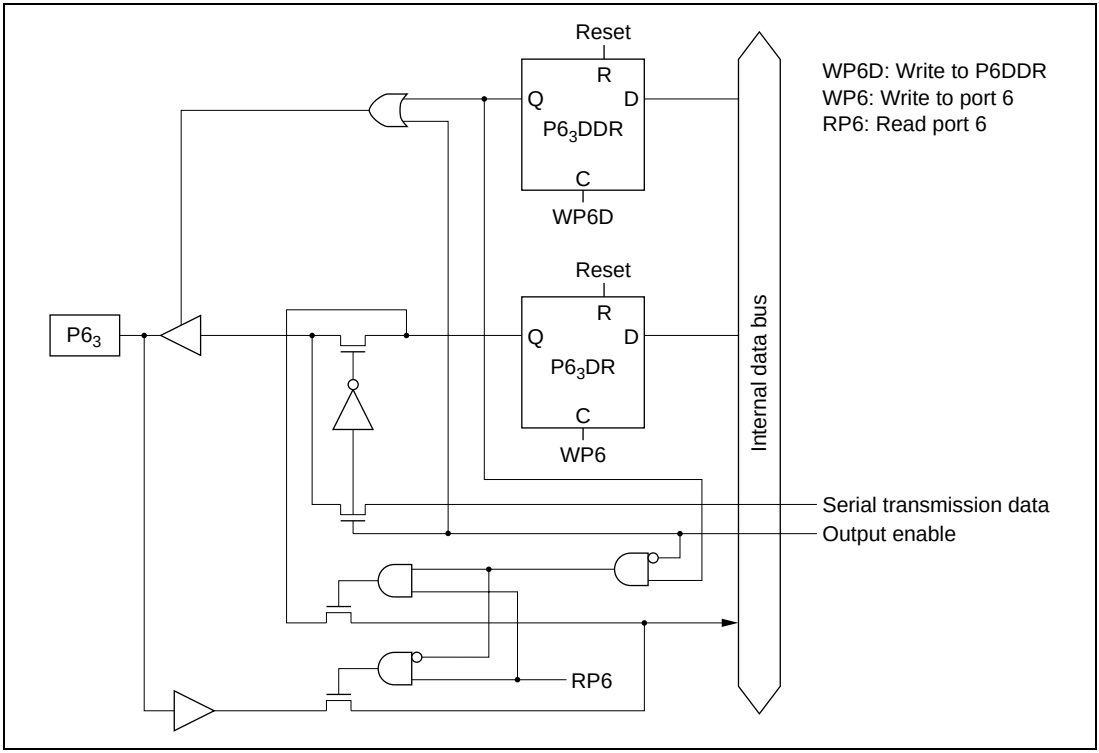


Figure C.3 (b) Schematic Diagram of Port 6, Pin P6$_3$

Table C.3 (b) Port 6 Port Read (Pin P6$_3$)

Setting		Port Read Data
Output enable		Pin value
Output disable	DDR = 0	Pin value
	DDR = 1	DR value

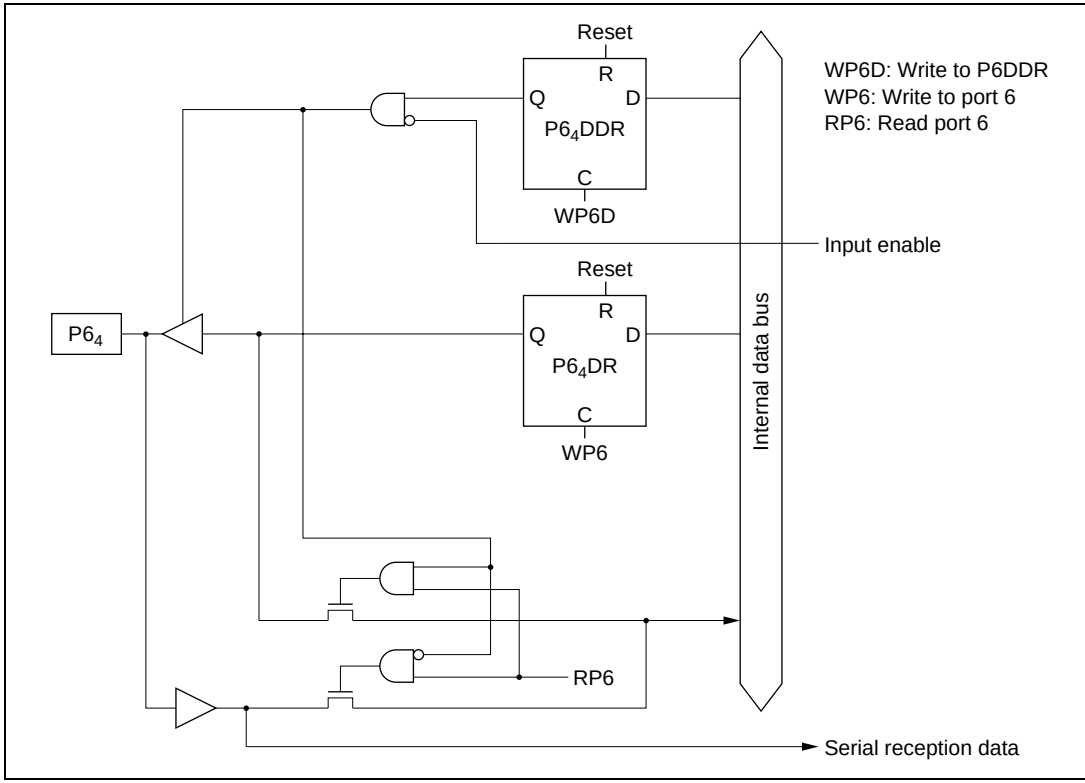


Figure C.3 (c) Schematic Diagram of Port 6, Pin P6$_4$

Table C.3 (c) Port 6 Port Read (Pin P6$_4$)

Setting		Port Read Data
Input enable		Pin value
Input disable	DDR = 0	Pin value
	DDR = 1	DR value

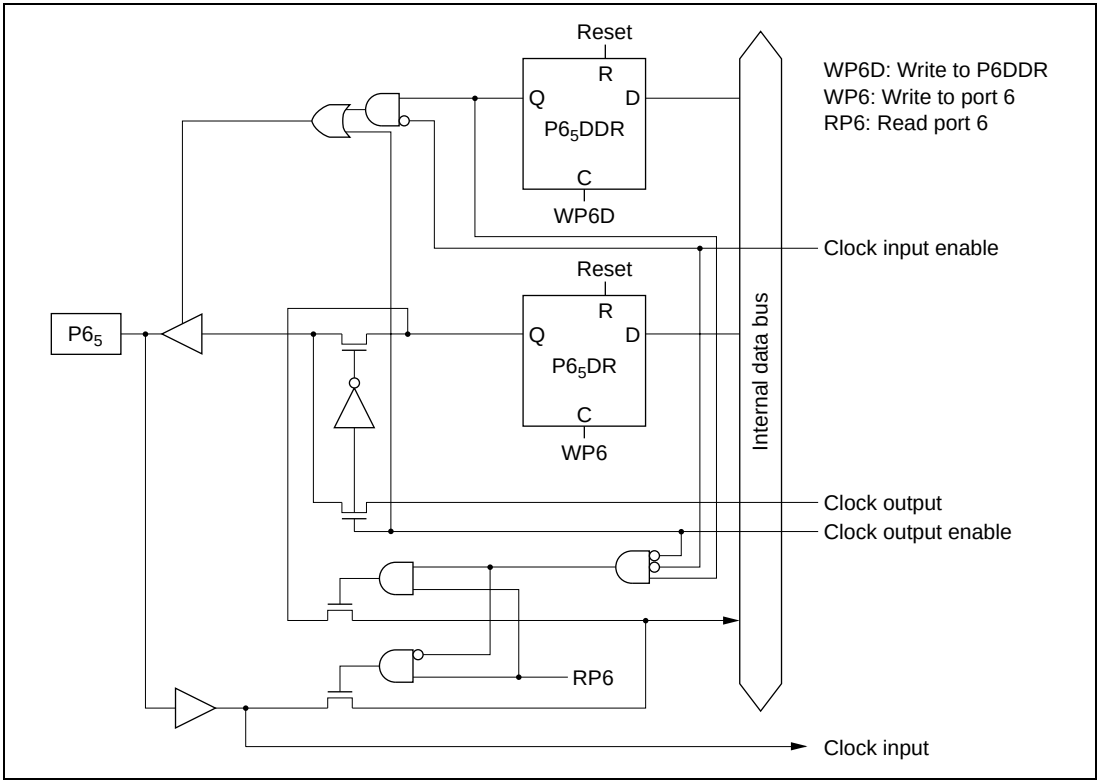


Figure C.3 (d) Schematic Diagram of Port 6, Pin P6$_5$

Table C.3 (d) Port 6 Port Read (Pin P6$_5$)

Setting		Port Read Data
Clock input enable		Pin value
Clock output enable		Pin value
Clock input/output disable	DDR = 0	Pin value
	DDR = 1	DR value

C.4 Schematic Diagram of Port 7

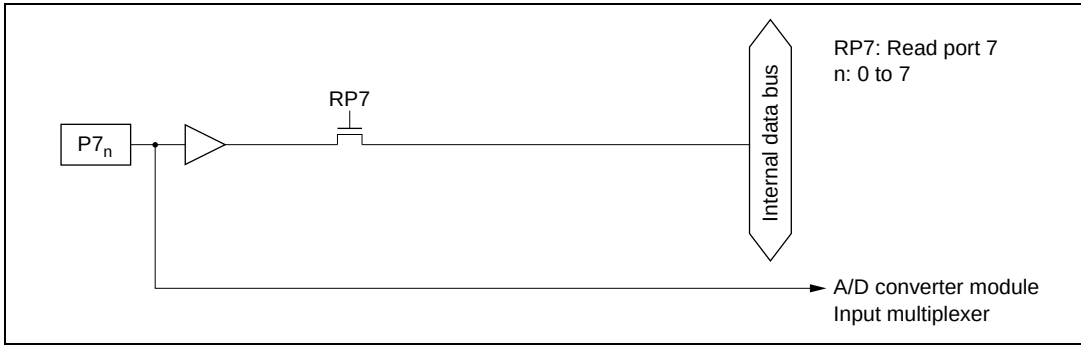


Figure C.4 Schematic Diagram of Port 7

C.5 Schematic Diagram of Port 8

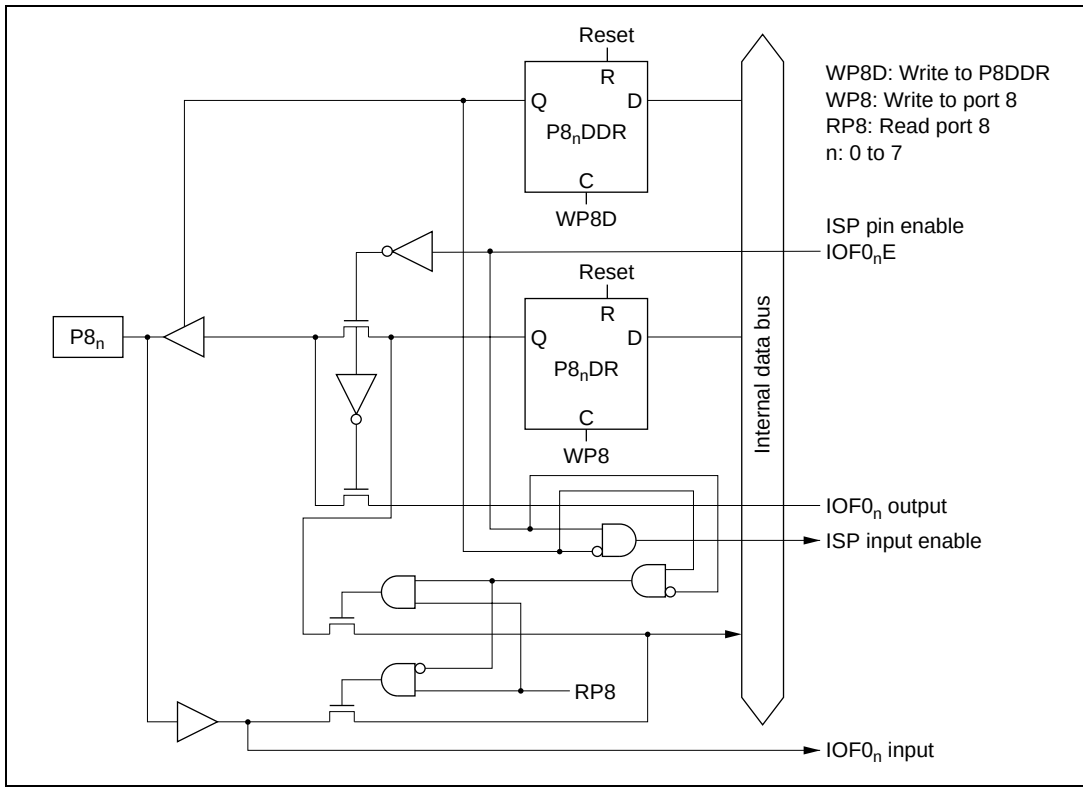


Figure C.5 Schematic Diagram of Port 8

Table C.5 Port 8 Port Read

Setting		Port Read Data
ISP pin enable IOF0$_n$E = 1	DDR = 0 (ISP input pin)	Pin value
	DDR = 1 (ISP output pin)	Pin value
ISP pin disable IOF0$_n$E = 0	DDR = 0 (Input port)	Pin value
	DDR = 1 (Output port)	DR value

C.6 Schematic Diagram of Port 9

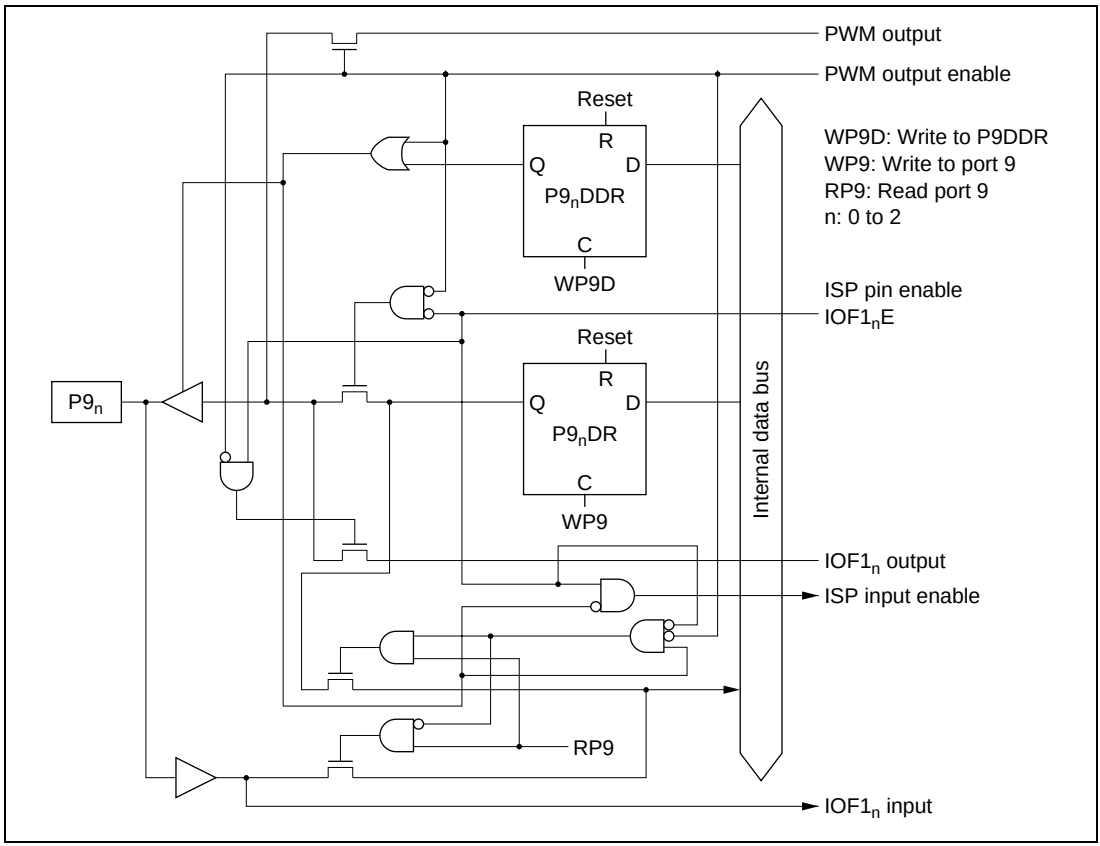


Figure C.6 (a) Schematic Diagram of Port 9, Pins $P9_0$, $P9_1$, and $P9_2$

Table C.6 (a) Port 9 Port Read (Pins $P9_0$, $P9_1$, and $P9_2$)

Setting			Port Read Data
PWM output disable	ISP pin enable $IOF0_nE$ = 1	DDR = 0 (ISP input pin)	Pin value
		DDR = 1 (ISP output pin)	Pin value
	ISP pin disable $IOF0_nE$ = 0	DDR = 0 (Input port)	Pin value
		DDR = 1 (Output port)	DR value
PWM output enable	—	—	Pin value

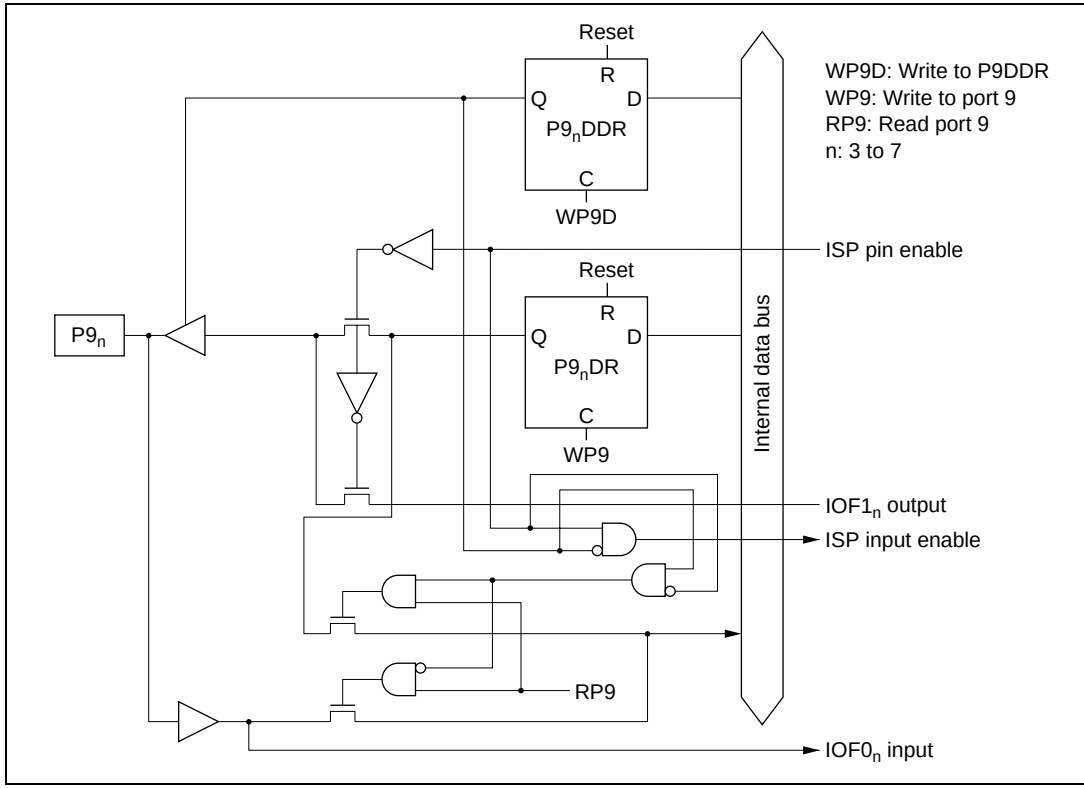


Figure C.6 (b) Schematic Diagram of Port 9, Pins $P9_3$ to $P9_7$

Table C.6 (b) Port 9 Port Read (Pins $P9_3$ to $P9_7$)

Setting		Port Read Data
ISP pin enable	DDR = 0 (ISP input pin)	Pin value
	DDR = 1 (ISP output pin)	Pin value
ISP pin disable	DDR = 0 (Input port)	Pin value
	DDR = 1 (Output port)	DR value

C.7 Schematic Diagram of Port 10

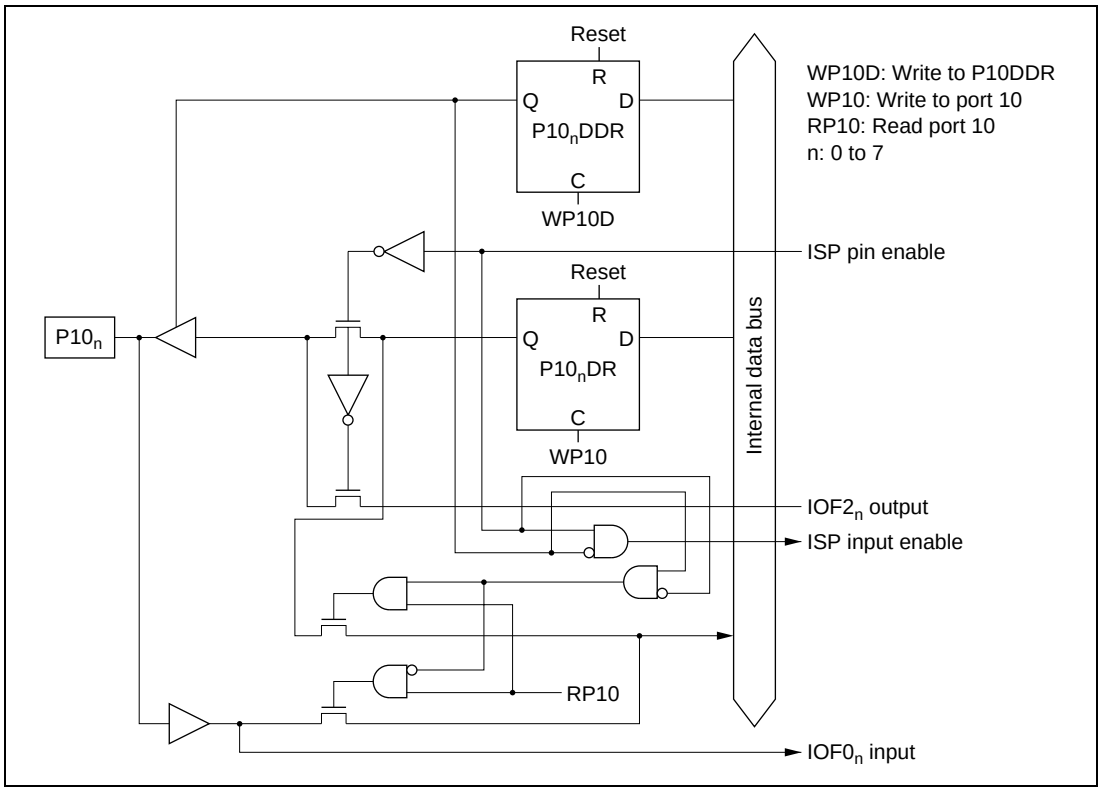


Figure C.7 Schematic Diagram of Port 10

Table C.7 Port 10 Port Read

Setting	Port Read Data
DDR = 0	Pin value
DDR = 1	DR value

C.8 Schematic Diagram of Port 11

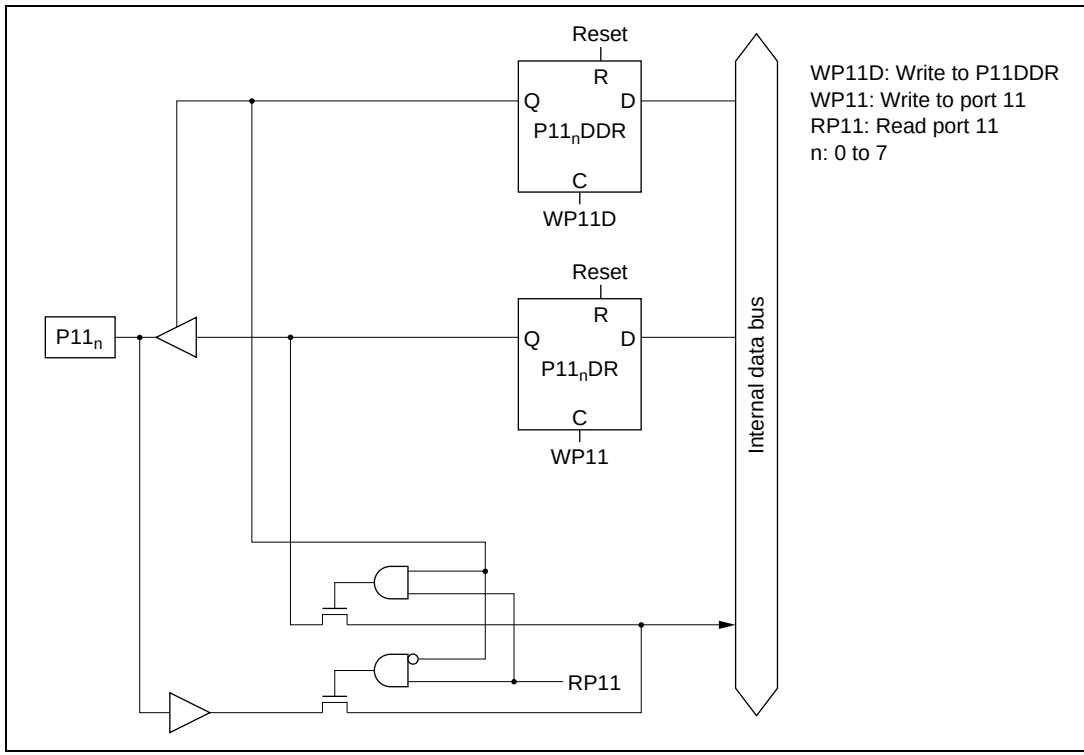


Figure C.8 Schematic Diagram of Port 11

Table C.8 Port 11 Port Read

Setting	Port Read Data
DDR = 0	Pin value
DDR = 1	DR value

C.9 Schematic Diagram of Port 12

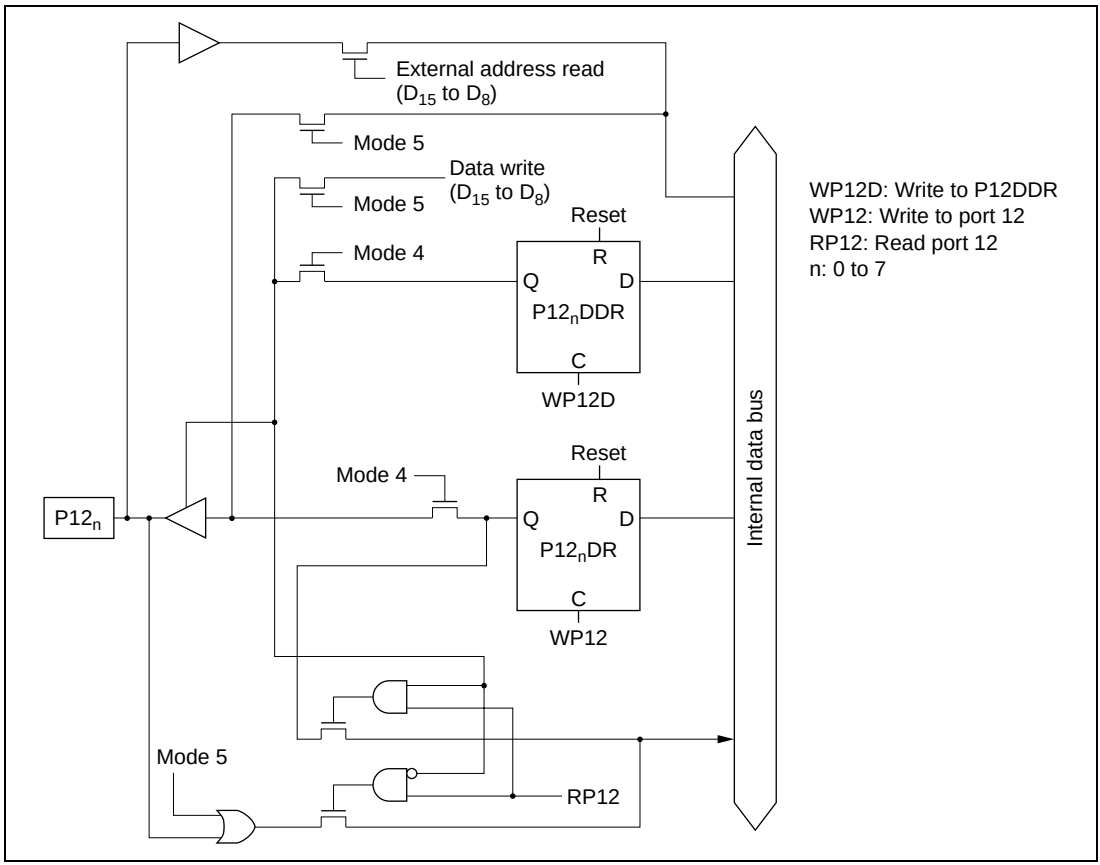
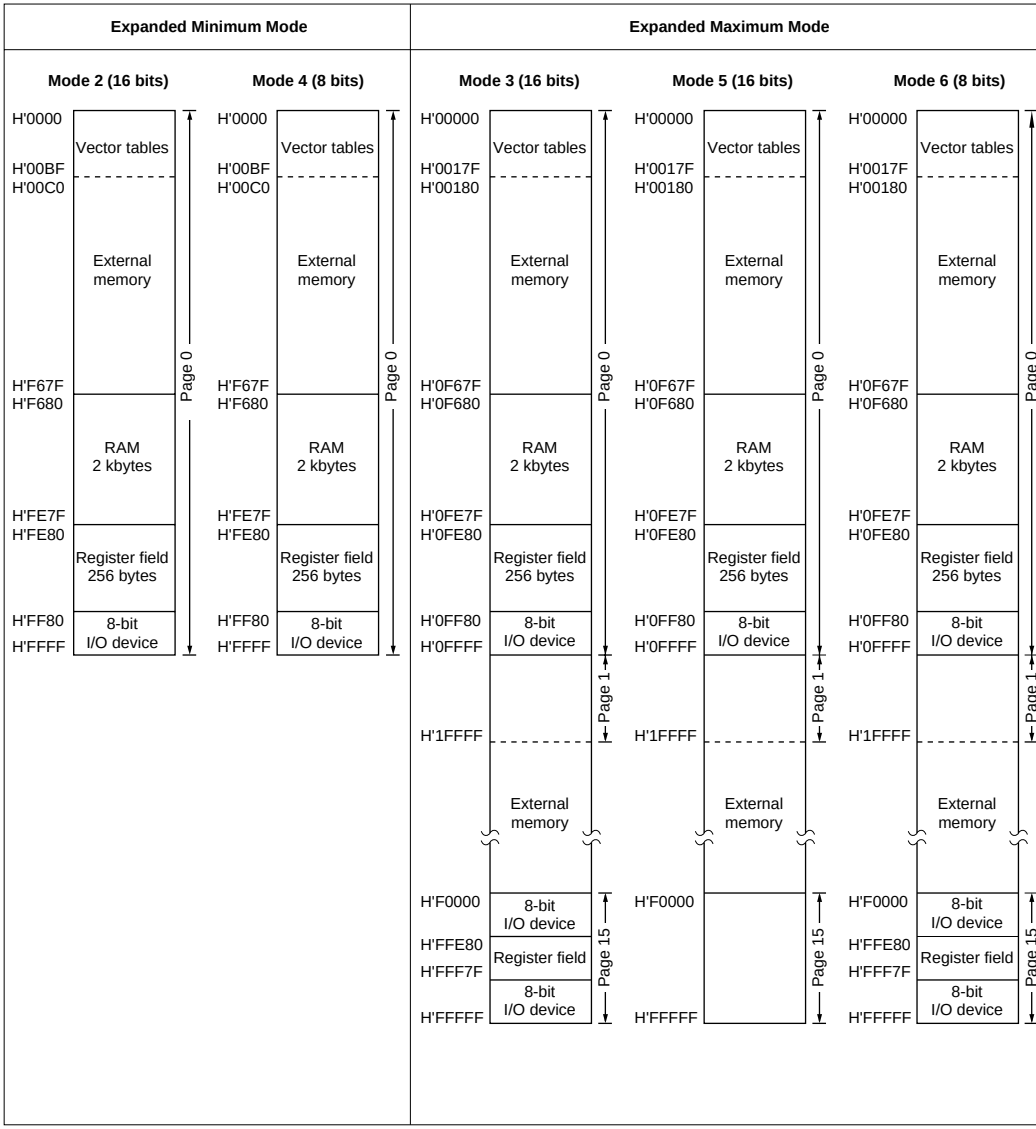


Figure C.9 Schematic Diagram of Port 12

Table C.9 Port 12 Port Read

Setting		Port Read Data
Mode 4	DDR = 0	Pin value
	DDR = 1	DR value
Mode 5		Always read as 1

Appendix D Memory Map



Appendix E Pin State

E.1 Port State of Each Pin State

Table E.1 Port State

Port Pin Name	Mode	Reset	Hardware Standby Mode	Software Standby Mode	Sleep Mode (Except ISP Memory Access)	Bus-Access-Right Release Mode	ISP Memory Access and Program Execution State (Normal Operation)
$P1_7$/$\overline{LWR}$	1	High	T	T	High	T	Control signal output
	3						
	4	T		Idle	Idle	Idle	Input/output port
	5	High		T	High	T	Control signal output
	6	T		Idle	Idle	Idle	Input/output port
$P1_3$/$\overline{WAIT}$ $P1_2$/$\overline{BREQ}$ $P5_4$/$\overline{IRQ_0}$	1	T	T	Idle	Idle*[1]	Idle	Input/output port or control signal input
	3						
	4						
	5						
	6						
$P1_1$/$\overline{BACK}$	1	T	T	Idle*[2]	Idle ($\overline{BACK}$ = high)	Idle ($\overline{BACK}$ = low)	Input/output port or control signal input
	3						
	4						
	5						
	6						
$P1_0$/ø	1	Clock output	T	(DDR = 1) ø = high (DDR = 0) T	(DDR = 1) Clock output (DDR = 0) T	(DDR = 1) Clock output (DDR = 0) T	(DDR = 1) Clock output (DDR = 0) Input port
	3						
	4						
	5						
	6						
$P5_6$/E	1	Clock output	T	(DDR = 1) E = low (DDR = 0) T	(DDR = 1) Clock output (DDR = 0) T	(DDR = 1) Clock output (DDR = 0) T	(DDR = 1) Clock output (DDR = 0) Input port
	3						
	4						
	5						
	6						

Port Pin Name	Mode	Reset	Hardware Standby Mode	Software Standby Mode	Sleep Mode (Except ISP Memory Access)	Bus-Access-Right Release Mode	ISP Memory Access and Program Execution State (Normal Operation)
$P5_3/A_{19}$ to $P5_0/A_{16}$	1	T	T	Idle	Idle	Idle	Input/output port
	3	Low		T	Low	T	A_{19} to A_{16}
	4	T		Idle	Idle	Idle	Input/output port
	5	Low		T	Low	T	A_{19} to A_{16}
	6						
$P5_7/\overline{ADTRG}$ $P6_2/PW_2$ to $P6_0/PW_0$, $P6_3$/TXD, $P6_4$/RXD, $P6_5$/SCK	1	T	T	Idle*[3]	Idle	Idle	Input/output port
	3						
	4						
	5						
	6						
$P7_7/AN_7$ to $P7_0/AN_0$	1	T	T	T	T	T	Input port
	3						
	4						
	5						
	6						
$P8_7/IOF0_7$ to $P8_0/IOF0_0$, $P9_7/IOF1_7$ to $P9_3/IOF1_3$, $P10_7/IOF2_7$ to $P10_0/IOF2_0$	1	T	T	Idle	Idle	Idle	Input/output port
	3						
	4						
	5						
	6						
$P9_2/PW_5$ $P9_1/PW_4$ $P9_0/PW_3$	1	T	T	Idle*[4]	Idle	Idle	Input/output port
	3						
	4						
	5						
	6						
$P11_7$ $P11_6$	1	T	T	Idle	Idle	Idle	Input/output port
	3						
	4						
	5						
	6						

Port Pin Name	Mode	Reset	Hardware Standby Mode	Software Standby Mode	Sleep Mode (Except ISP Memory Access)	Bus-Access-Right Release Mode	ISP Memory Access and Program Execution State (Normal Operation)
$P12_7/D_{15}$ to $P12_0/D_8$	1	T	T	T	T	T	D_{15} to D_8
	3						
	4			Idle	Idle	Idle	Input/output port
	5			T	T	T	D_{15} to D_8
	6			Idle	Idle	Idle	Input/output port
$\overline{AS}$, $\overline{RD}$, $\overline{WR}$ ($\overline{HWR}$)	1	High	T	T	High	T	Control signal output
	3						
	4						
	5						
	6						
A_{15} to A_0	1	Low	T	T	Low	T	Address output
	3						
	4						
	5						
	6						
D_7 to D_0	1	T	T	T	T	T	Data bus
	3						
	4						
	5						
	6						

Notes: T: High impedance

Idle: For input port, high-impedance state; for output port, maintains its previous state.

1. $\overline{BREQ}$ can be received.
2. If $P1_1$ is set for $\overline{BACK}$ output, $P1_1$ goes into high-impedance state.
3. The on-chip supporting modules are reset, so these pins become initialized as input or output pins according to their DDR and DR bits.
4. The PWM timer is reset, so these pins become initialized as input or output pins or ISP pins according to their DDR, DR, and SYSCR9 bits.

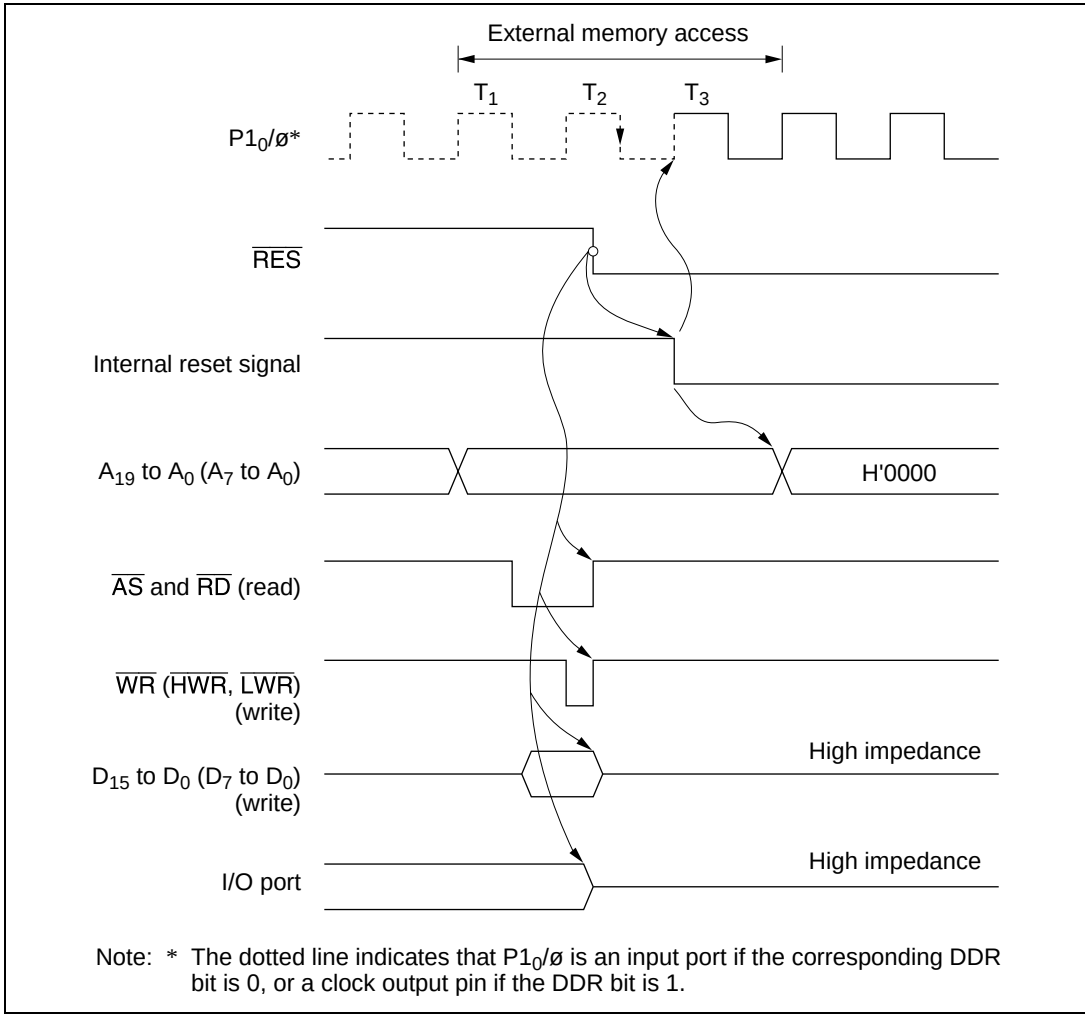
E.2 Pin Status in the Reset State

Reset during Three-State Access: Pin states change accordingly when the $\overline{\text{RES}}$ pin is asserted low during three-state access of external memory. See figure E.1.

As soon as $\overline{\text{RES}}$ goes low, all ports are initialized to input ports. The $\overline{\text{AS}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ ($\overline{\text{HWR}}$ and $\overline{\text{LWR}}$) signals all go high. The data bus D_{15} to D_0 (D_7 to D_0) is placed in the high-impedance state.

The address bus is initialized 1.5ø clock periods after the low input at the $\overline{\text{RES}}$ pin is sampled. All address bus signals are driven low.

The clock output pins $P1_0$/ø and $P5_5$/E are also initialized 0.5ø clock periods after the low input at the $\overline{\text{RES}}$ pin is sampled. Both pins are initialized as output pins.



Note: * The dotted line indicates that $P1_0/ø$ is an input port if the corresponding DDR bit is 0, or a clock output pin if the DDR bit is 1.

Figure E.1 Reset During Memory Access (During Three-State Access)

Reset during Two-State Access: Pin states change accordingly when the $\overline{RES}$ pin is asserted low during two-state access of external memory. See figure E.2.

As soon as $\overline{RES}$ goes low, all ports are initialized as input ports. The $\overline{AS}$, $\overline{RD}$, and $\overline{WR}$ ($\overline{HWR}$ and $\overline{LWR}$) signals all go high. The data bus D_{15} to D_0 (D_7 to D_0) is placed in the high-impedance state.

The address bus is initialized 1.5ø clock periods after the low input at the $\overline{RES}$ pin is sampled. All address bus signals are driven low.

The clock output pins $P1_0$/ø and $P5_5$/E are also initialized 0.5ø clock periods after the low input at the $\overline{RES}$ pin is sampled. Both pins are initialized as output pins.

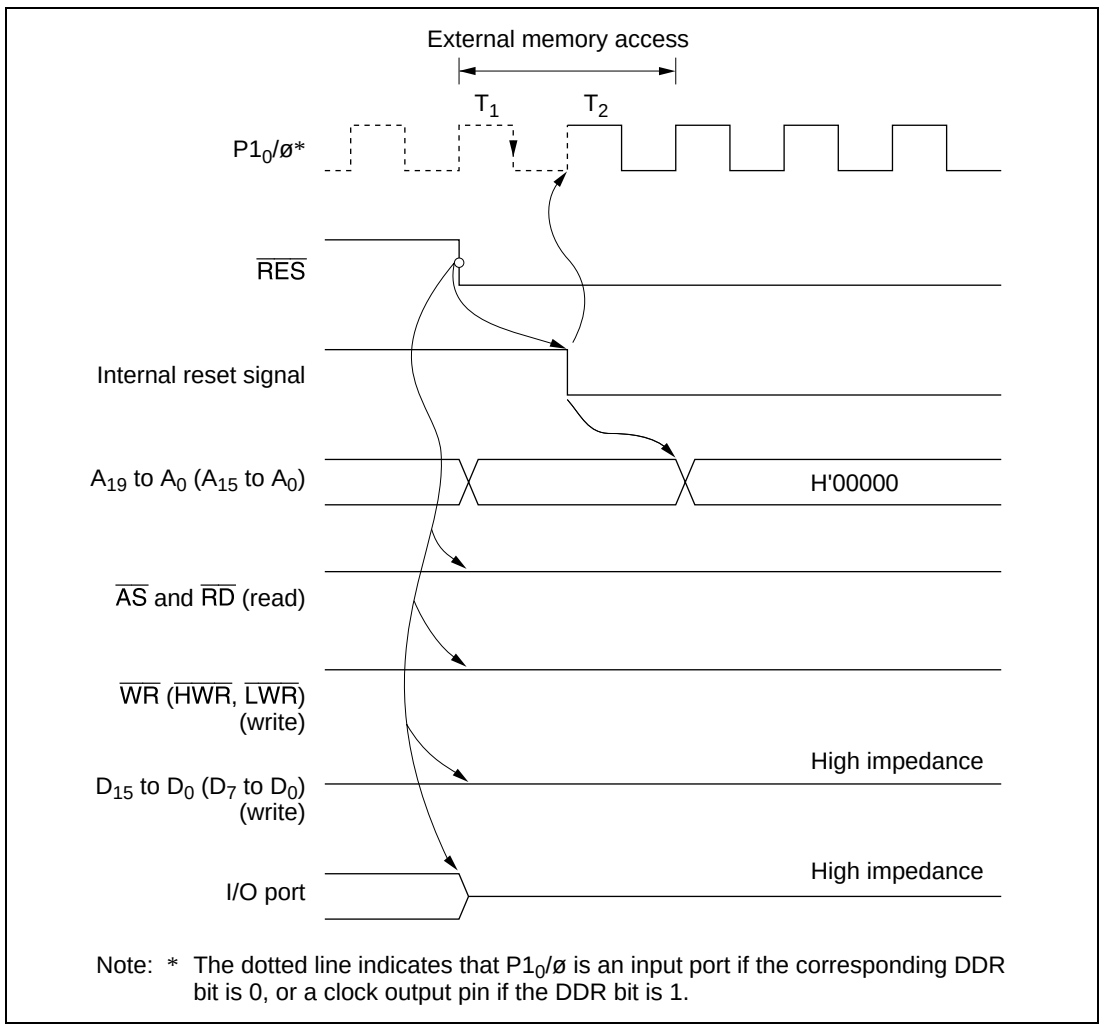
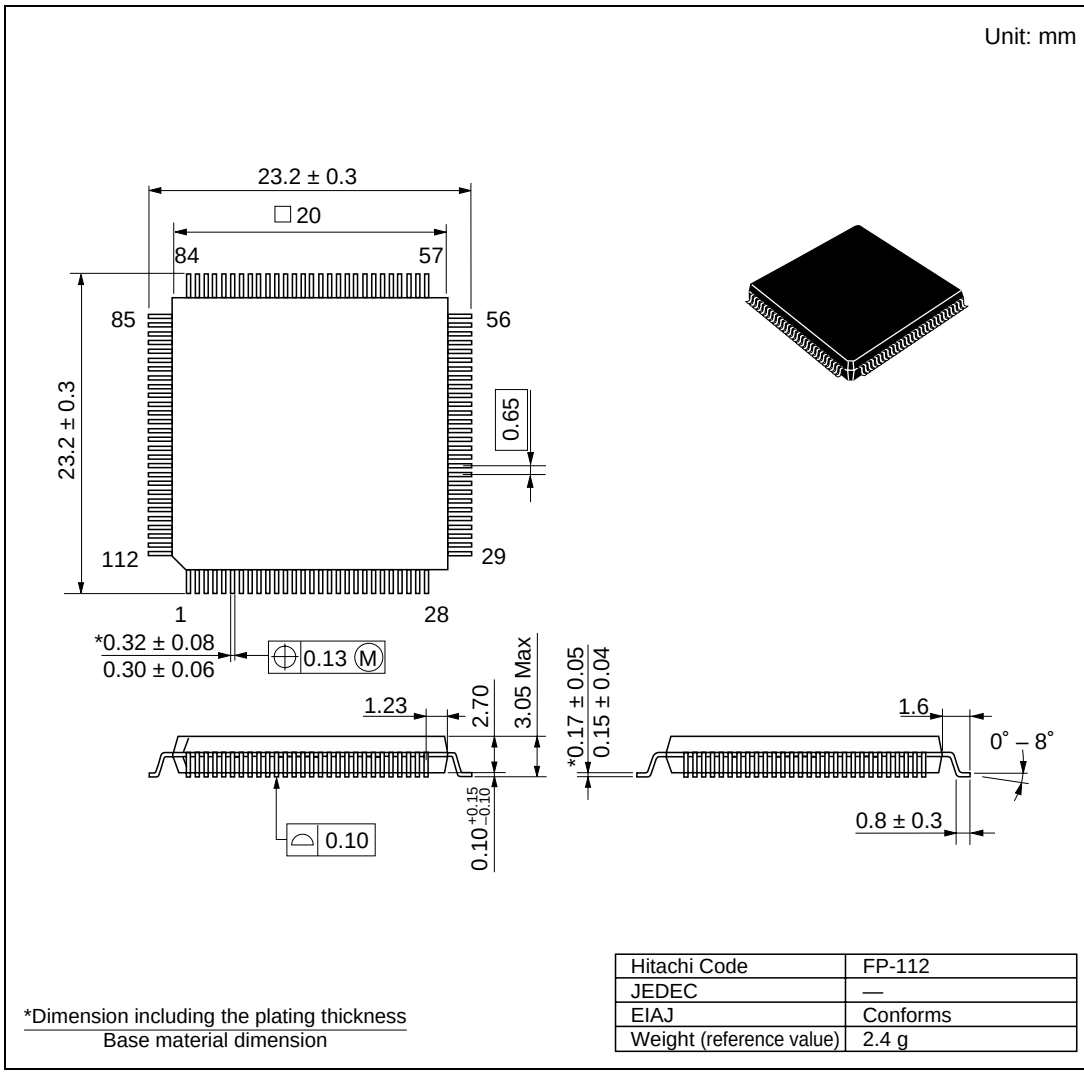

External memory access
T1
T2
P1₀/ø*
RES
Internal reset signal
A19 to A0 (A15 to A0)
H'00000
AS and RD (read)
WR (HWR, LWR) (write)
High impedance
D15 to D0 (D7 to D0) (write)
High impedance
I/O port
Note: * The dotted line indicates that P1₀/ø is an input port if the corresponding DDR bit is 0, or a clock output pin if the DDR bit is 1.

Figure E.2 Reset During Memory Access (During Two-State Access)

Appendix F Package Dimensions



Hitachi Code	FP-112
JEDEC	—
EIAJ	Conforms
Weight (reference value)	2.4 g

Figure F.1 Package Dimensions (QFP-112)

Appendix G Abbreviations Listing

#Imm: Immediate data

(EAd): Destination operand

(EAs): Source operand

ACIA: Asynchronous communication interface adapter

AD_9–AD_0: A/D conversion result bits

ADCR: A/D control register

ADCSR: A/D control/status register

ADDRA–ADDRD: A/D data registers

ADF: A/D end flag bit

ADI: A/D conversion completion interrupt

ADIE: A/D interrupt enable bit

ADST: A/D start bit

$\overline{\text{ADTRG}}$: External trigger input pin

AI_{CC}: Analog supply current

AN_7–AN_0: Analog input pins

$\overline{\text{AS}}$: Address strobe output pin

AV_{CC}: Analog reference voltage input pin

AV_{SS}: Analog ground input pin

B: Byte

$\overline{\text{BACK}}$: Bus request acknowledge output pin

BCC1–BCC0: Bus cycle control bits

BR: Base register

$\overline{\text{BREQ}}$: Bus request input pin

BRLE: Bus release enable bit

BRR: Bit rate register

C/$\overline{\text{A}}$: Communication mode bit

C: Carry bit

CCR: Condition code register

$\overline{\text{CE}}$: EPROM chip enable pin

CH2–CH0: Channel select bits

CHR: Character length bit

C_{in}: Input capacitance

CKE1, CKE0: Clock enable bits

CKS2–CKS0: Clock select bits

CLEH: Read clear enable register high register

CLEL: Read clear enable register low register

CP: Code page register

CPG: Clock pulse generator

CR: Control register

DI: Destination increment

disp: Displacement

DP: Data page register

DRn: Data register n

DTC: Data transfer controller

DTCR: Data transfer count register

DTDR: Data transfer destination address register

DTEA–DTED: Data transfer enable registers

DTMR: Data transfer mode register

DTSR: Data transfer source address register

E: Enable clock output pin

EA_{16}–EA_0: EPROM address input pins

EGF: Edge detection flags register

EO_7–EO_0: EPROM data I/O pins

EP: Extended page register

ERI: Receive error interrupt

EVER: Event input enable register

EXTAL: External crystal input pin

FEDGE: Falling edge enable register

FER: Framing error bit

f_{OSC}: Oscillator frequency

FP: Frame pointer

$\overline{FRM}$: Free-running mode bit

$\overline{HWR}$: Upper byte write output pin

I_2–I_0: Interrupt mask bits

IBE: ISP bus enable bit

I_{CC}: Current dissipation

ICFH: Interconnection flags high register

ICFL: Interconnection flags low register

ICSR: ISP control status register

IEFH: Interrupt enable flags high register

IEFL: Interrupt enable flags low register

I_{in}: Input leakage current

INTC: Interrupt controller

IOF2–IOF0: I/O flags registers

$IOF0_7E$–$IOF0_0E$: $IOF0_7$–$IOF0_0$ enable bits

$IOF1_7E$–$IOF1_0E$: $IOF1_7$–$IOF1_0$ enable bits

$IOF2_7E$–$IOF2_0E$: $IOF2_7$–$IOF2_0$ enable bits

I_{OH}: Allowable output high current sink

IOIEH: I/O interrupt enable flags high register

IOIEL: I/O interrupt enable flags low register

I_{OL}: Allowable output low current sink

I_p: Input pull-up MOS current

IPRA–IPRD: Interrupt priority registers

$\overline{IRQ_0}$: Interrupt request 0 input pin

IRQ_0E: Interrupt request enable 0 bit

ISFH: Interrupt status flags high register

ISFL: Interrupt status flags low register

ISP: Intelligent Sub-Processor

I_{TSI}: Leakage current in tristate

LSB: Least significant bit

$\overline{LWR}$: Lower byte write output pin

MD_2–MD_0: Mode select input pins

M_D: Number of states for writing destination data

MDCR: Mode control register

MDS2–MDS0: Mode select bits

M_S: Number of states for reading source data

MSB: Most significant bit

N: Negative bit

NMI: Nonmaskable interrupt input pin

NMIEG: Nonmaskable interrupt edge bit

O/$\overline{E}$: Parity mode bit

OCF2–OCF0: Output compare flag bits

OCI2–OCI0: Output compare interrupts

OCIE2–OCIE0: Output compare interrupt enable bits

OCR2–OCR0: Output compare registers

ODL: Output data latch

ODR2–ODR0: Output data registers

OE2–OE0: Output enable bits

$\overline{OE}$: EPROM output enable pin

OMS: Output mode select bit

ORER: Overrun error bit

OVF: Overflow flag bit

P12DDR–P1DDR: Port data direction registers

P12DR–P1DR: Port data registers

P12–P1: Parallel I/O pins

$P1_3$–$P1_0$: Port 1 data register bits

$P1_7$DDR, $P1_3$DDR–$P1_0$DDR: Port 1 data direction register bits

$P5_7$DDR, $P5_6$DDR, $P5_4$DDR–$P5_0$DDR: Port 5 data direction register bits

$P5_7$–$P5_6$, $P5_4$–$P5_0$: Port 5 data register bits

$P6_5$DDR–$P6_0$DDR: Port 6 data direction register bits

$P6_5$–$P6_0$: Port 6 data register bits

$P7_7$–$P7_0$: Port 7 data register bits

$P8_7DDR–P8_0DDR$: Port 8 data direction register bits

$P8_7–P8_0$: Port 8 data register bits

$P9_7DDR–P9_0DDR$: Port 9 data direction register bits

$P9_7–P9_0$: Port 9 data register bits

$P10_7DDR–P10_0DDR$: Port 10 data direction register bits

$P10_7–P10_0$: Port 10 data register bits

$P11_7DDR–P11_0DDR$: Port 11 data direction register bits

$P11_7–P11_0$: Port 11 data register bits

$P12_7DDR–P12_0DDR$: Port 12 data direction register bits

$P12_7–P12_0$: Port 12 data register bits

PC: Program counter

PE: Parity enable bit

PER: Parity error bit

$\overline{PGM}$: EPROM programming enable pin

$PW_5–PW_0$: PWM timer output pins

PWM: Pulse width modulation

R7–R0: General registers

RAMCR: RAM control register

RAME: RAM enable bit

Rd: General destination register

RDR: Receive data register

RDRF: Receive data register full bit

RE: Receive enable bit

REDGE: Rising edge enable register

$\overline{RES}$: Reset I/O pin

RI: Receive interrupt bit

RIE: Receive interrupt enable bit

Rn: General register n

rrr: Register number field bits

Rs: General source register

RSR: Receive shift register

RSTCSR: Reset control/status register

RSTOE: Reset output enable bit

RXD: SCI receive data input pin

RXI: Receive-end interrupt

SBYCR: Software standby control register

SCAN: Scan mode bit

SCI: Serial communication interface

SCK: SCI clock I/O pin

SCR: Serial control register

SI: Source increment

SMR: Serial mode register

SP: Stack pointer

SR: Status register

SSBY: Software standby bit

SSR: Serial status register

$\overline{\text{STBY}}$: Standby input pin

STOP: Stop bit length bit

SYSCR1: System control register 1

SYSCR8: System control register 8

SYSCR9: System control register 9

SYSCR10: System control register 10

Sz: Size

t_{ACC1}, t_{ACC2}: Read data access times

t_{AD}: Address delay time

t_{AH}: Address hold time

t_{AS1}–t_{AS3}: Address setup times

t_{ASD1}, t_{ASD2}: $\overline{\text{AS}}$ output delay times

t_{BACD1}, t_{BACD2}: Bus acknowledge delay times

t_{BRQS}: Bus request setup time

t_{BZD}: Bus floating delay time

TCE: Timer count enable bit

t_{Cf}: Clock fall time

t_{CH}: Clock pulse width high

t_{CL}: Clock pulse width low

TCNT: Timer counter

t_{CONV}: Total A/D conversion time

TCR: Timer control register

t_{Cr}: Clock rise time

TCSR: Timer control/status register

t_{cyc}: Clock cycle time

t_{D}: Synchronization delay time

TDR: Transmit data register

TDRE: Transmit data register empty bit

t_{DSWW}: Write data strobe pulse width

TE: Transmit enable bit

t_{ED}: E-clock delay time

t_{Ef}: E-clock fall time

TEMP: Temporary register

t_{Er}: E-clock rise time

TIE: Transmit interrupt enable bit

t_{IRQ0S}: IRQ_0 setup time

t_{ISPD}: ISP output data delay time

t_{ISPH}: ISP input data hold time

t_{ISPS}: ISP input data setup time

TME: Timer enable bit

TMR: 16-bit timer

TMSR: Timer status register

t_{NMIH}: NMI hold time

t_{NMIS}: NMI setup time

T_{opr}: Operating temperature

t_{OSC1}: Crystal oscillator settling time

T_p: Precharge state

TP: Stack page register

t_{PRH}: Port input data hold time

t_{PRS}: Port input data setup time

t_{PWD}: Port output data delay time

t_{PWOD}: PWM timer output delay time

t_{RDD1}, t_{RDD2}: $\overline{RD}$ output delay times

t_{RDH}: Read data hold time

t_{RDHE}: Read data hold time (E-clock sync)

t_{RDS}: Read data setup time

t_{RDSE}: Read data setup time (E-clock sync)

TRE2, TRE0: Transfer request enable bits

t_{RESD}: $\overline{RES}$ output delay time

t_{RESOW}: $\overline{RES}$ output pulse width

t_{RESS}: $\overline{RES}$ setup time

t_{RESW1}, t_{RESW2}: $\overline{RES}$ pulse widths

TRGE: Trigger enable bit

t_{RXH}: SCI receive data hold time

t_{RXS}: SCI receive data setup time

t_{SCKW}: SCI input clock pulse width

t_{Scyc}: SCI input clock cycle

t_{SPL}: Input sampling time

TSR: Transmit shift register

T_{stg}: Storage temperature

t_{TXD}: SCI transmit data delay time

T_w: Wait state

t_{WDD}: Write data delay time

t_{WDS}: Write data setup time

t_{WRD1}–t_{WRD3}: $\overline{WR}$ output delay times

t_{WTH}: $\overline{WAIT}$ delay time

t_{WTS}: $\overline{WAIT}$ setup time

TXD: SCI transmit data output pin

TXI: Transmit-end interrupt

UART: Universal asynchronous receiver/ transmitter

V: Overflow bit

V_{IH}: Input high voltage

V_{IL}: Input low voltage

V_{in}: Input voltage

V_{OH}: Output high voltage

V_{OL}: Output low voltage

V_{PP}: EPROM programming power supply pin

V_{RAM}: RAM standby voltage

W: Word

WC1–WC0: Wait count bits

WDT: Watchdog timer

WMS1–WMS0: Wait mode select bits

WRST: Watchdog timer reset bit

WSC: Wait state controller

WT/$\overline{IT}$: Timer mode select bit

XTAL: Crystal input pin

Z: Zero bit

ZTAT™: Zero Turn Around Time

H8/570 Hardware Manual

Publication Date: 1st Edition, March 1992
Published by: Electronic Devices Sales & Marketing Group
Semiconductor & Integrated Circuits
Hitachi, Ltd.
Edited by: Technical Documentation Group
Hitachi Kodaira Semiconductor Co., Ltd.
Copyright © Hitachi, Ltd., 1992. All rights reserved. Printed in Japan.